Numerical modelling and High Performance Computing for sediment flows: Part two

Jean-Baptiste Keck

Thursday 7th February, 2019

Table of contents

Coastal sedimentary processes

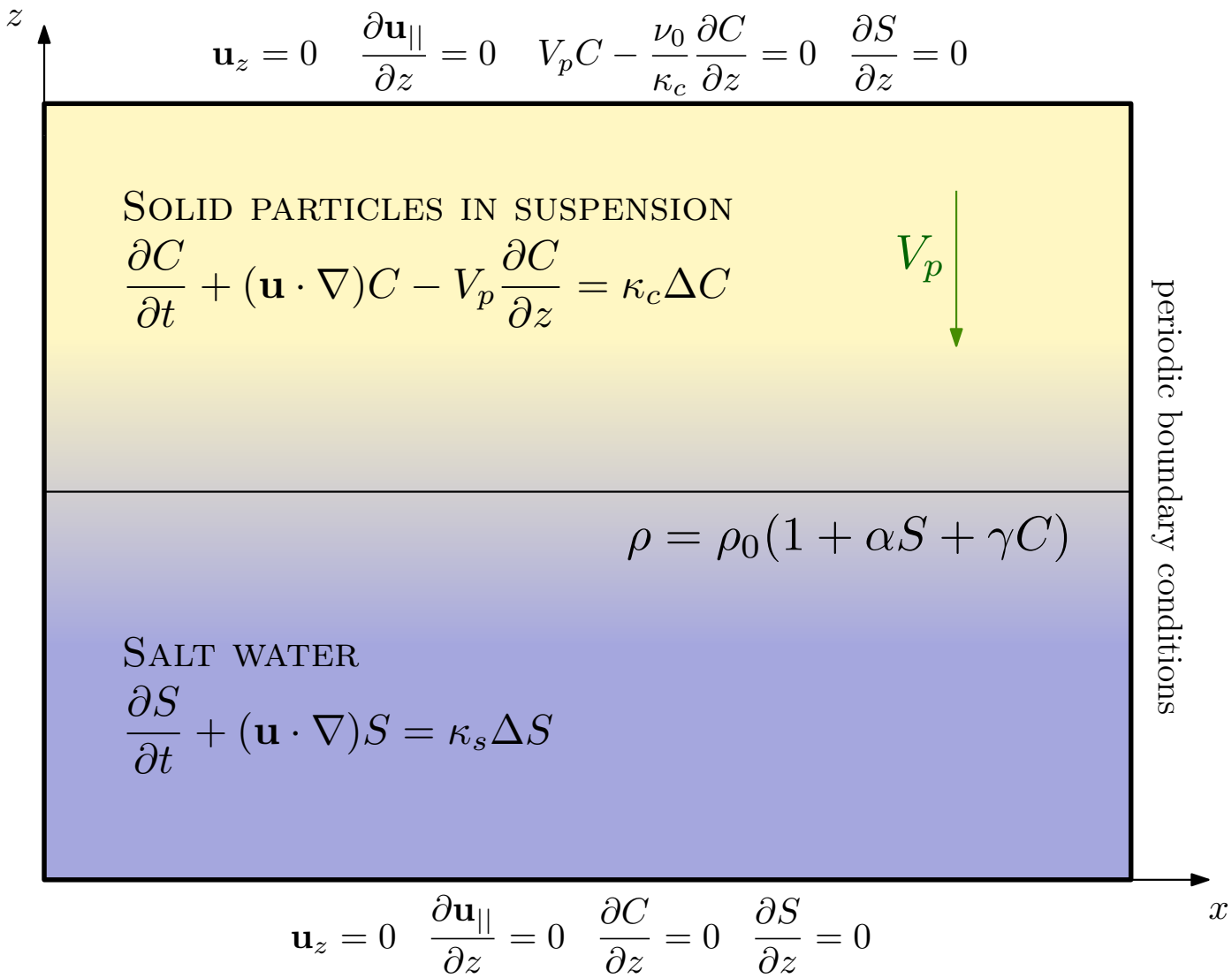
Physics of particle-laden fresh water flows above salted water:

River delta of Irrawady, Myanmar (ESA)

Modelling sediment flows

Particle ladden fresh water above salt water

Considered numerical setup



We consider the same numerical setup as in [1].

Target numerical simulation

Given the following physical fields:

- Usual velocity field $\boldsymbol{u}(\boldsymbol{x}, t)$ and pressure field $P(\boldsymbol{x}, t)$.
- $0 \leq S(\boldsymbol{x}, t) \leq 1$ the salinity, a scalar field advected in the fluid.
- $0 \leq C(\boldsymbol{x}, t) \leq 1$ the particles concentration, another scalar field.

$$\rho = \rho_0 \left(1 + \alpha S + \gamma C\right) \tag{1a}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \boldsymbol{u} = 0 \tag{1b}$$

$$\rho \left[\frac{\partial \boldsymbol{u}}{\partial t} + \left(\boldsymbol{u} \cdot \nabla\right) \boldsymbol{u}\right] = \eta_0 \Delta \boldsymbol{u} - \boldsymbol{\nabla} P + \rho \boldsymbol{g} \tag{1c}$$

$$\frac{\partial S}{\partial t} + \left(\boldsymbol{u} \cdot \nabla\right) S = \kappa_s \Delta S \tag{1d}$$

$$\frac{\partial C}{\partial t} + \left(\boldsymbol{u} \cdot \nabla\right) C - V_p \frac{\partial C}{\partial z} = \kappa_c \Delta C \tag{1e}$$

- Navier-Stokes eqs. coupled with two scalar advection-diffusion eqs.

Boussinesq approximation

- As α and γ are small (3-4%) we can use the Boussineq approximation:

$$\rho = \rho_0 \left(1 + \alpha S + \gamma C\right) \tag{2a}$$

$$\frac{\partial \rho_0}{\partial t} + \nabla \cdot \rho_0 \boldsymbol{u} = 0 \tag{2b}$$

$$\rho_0 \left[\frac{\partial \boldsymbol{u}}{\partial t} + \left(\boldsymbol{u} \cdot \nabla\right) \boldsymbol{u}\right] = \eta_0 \Delta \boldsymbol{u} - \boldsymbol{\nabla} P + \rho \boldsymbol{g} \tag{2c}$$

$$\frac{\partial S}{\partial t} + \left(\boldsymbol{u} \cdot \nabla\right) S = \kappa_s \Delta S \tag{2d}$$

$$\frac{\partial C}{\partial t} + \left(\boldsymbol{u} \cdot \nabla\right) C - V_p \frac{\partial C}{\partial z} = \kappa_c \Delta C \tag{2e}$$

- The density variation is only taken into account in the buoyancy term.

Instabilities

Constants of the problem

Physical constants of the problem:

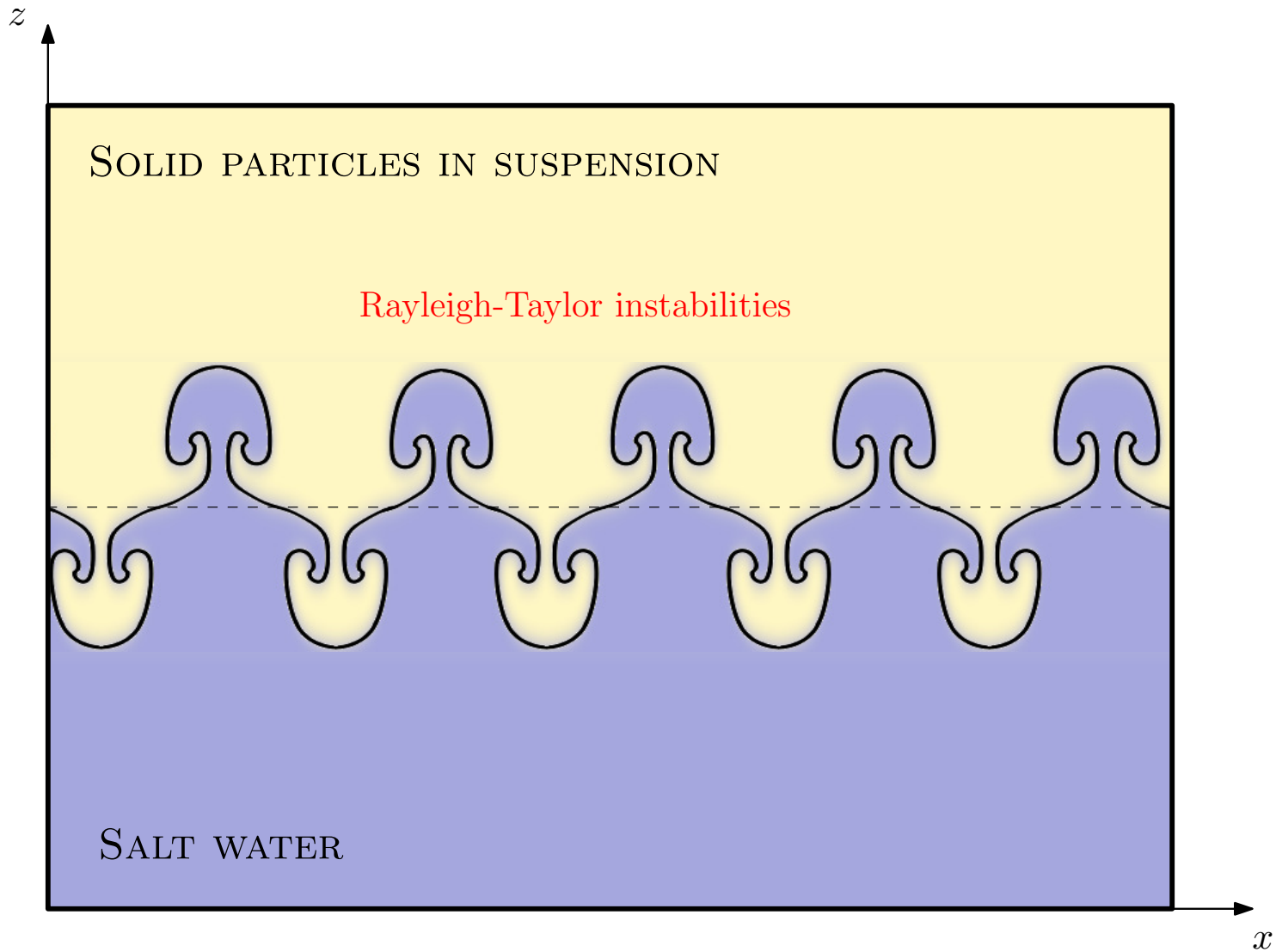
ρ_0	fresh water density
η_0	dynamic viscosity of fresh water
α, γ	density expansion coefficients of salinity and sediments
κ_s, κ_c	diffusion coefficients of salinity and sediments
V_{st}	Stokes settling velocity of the particles

Some dimensionless constants of the problem:

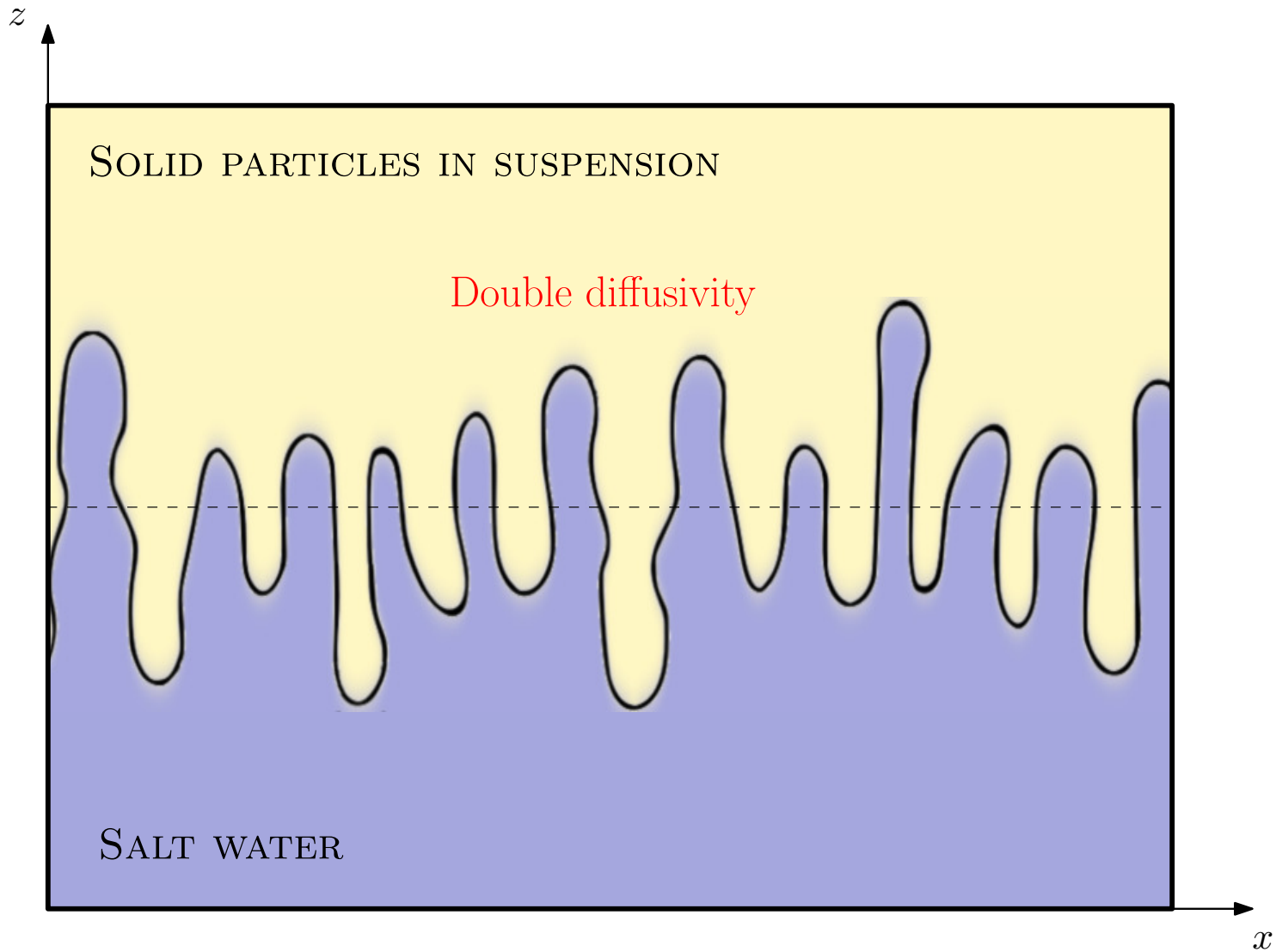
R_s	$=$	$\frac{\alpha}{\gamma}$	stability ratio
S_c	$=$	$\frac{\nu_0}{\kappa_s}$	Schmidt number
τ	$=$	$\frac{\kappa_s}{\kappa_c}$	diffusivity ratio
V_p	$=$	$\frac{V_{st}}{(\nu_0 g')^{\frac{1}{3}}}$	dimensionless settling velocity of the particles

Depending on those parameters, different regimes may appear.

Rayleigh-Taylor instabilities



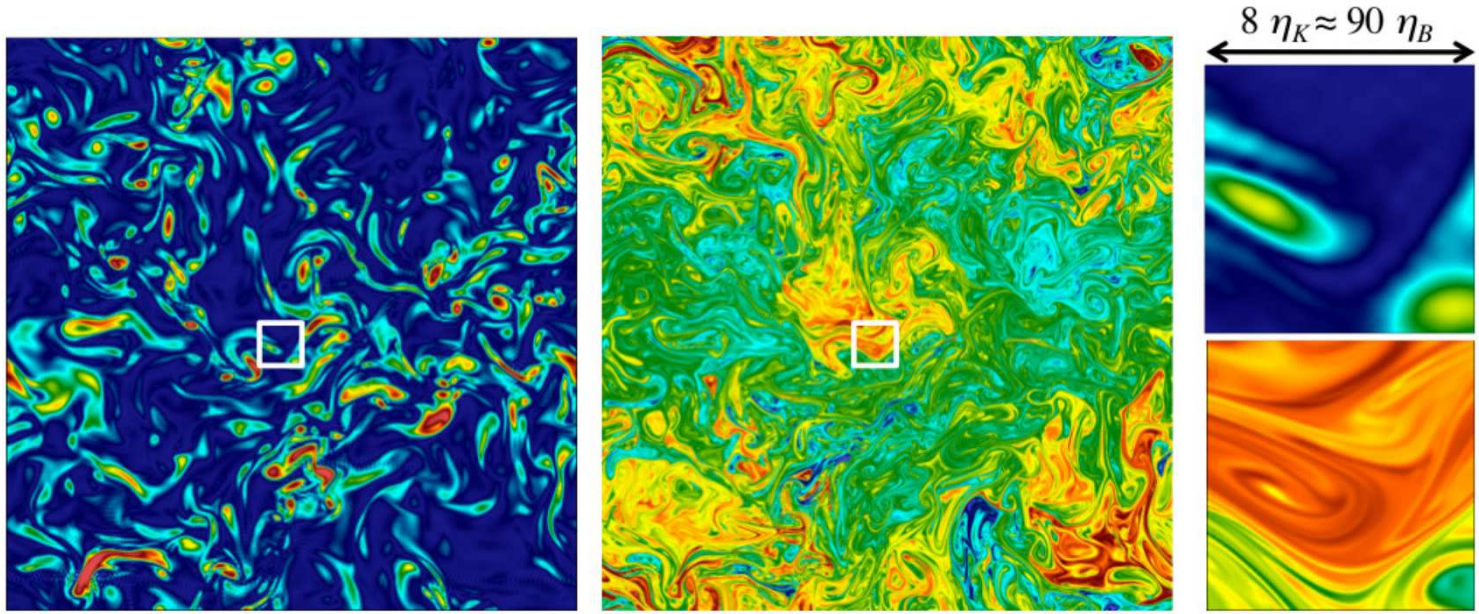
Double diffusivity



Numerical considerations

Numerical considerations

- The Schmidt number for salt and water is approximately $S_c = 700$.
- Assuming $\tau = 25$, this gives a Schmidt number of $\tau S_c = 17500$.
- This requires important computational ressources even in 2D.



- Even if we use a multiscale resolution strategy as shown in [1].
- The physics tells us that the ratio between velocity and scalar grids should be 27 and 133, in each direction !

Numerical considerations

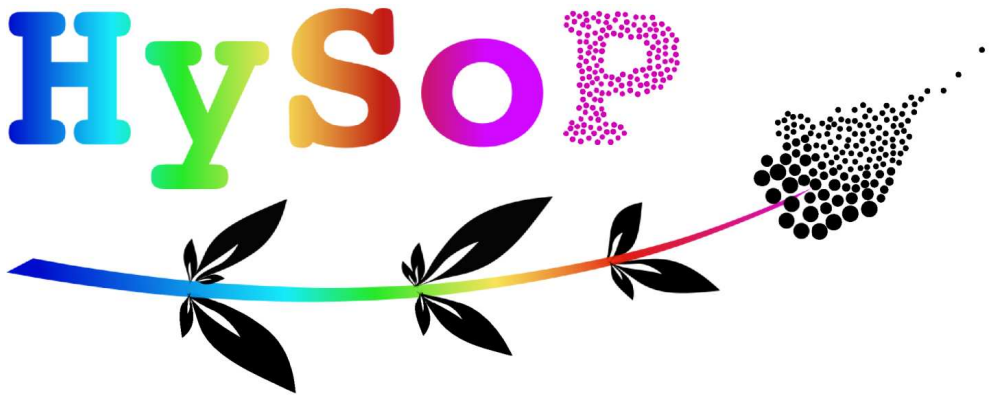
- Highest Schmidt number S_c achieved by Meiburg et al in [1] is:
  - 70 in 2D (2048×4097 grid).
  - 7 in 3D ($512 \times 512 \times 1537$ grid).

- It would be interesting to extend their results towards more physical cases (higher Schmidt numbers).

- We want to propose a new numerical approach featuring:
  - A fully distributed `MPI` simulation `[WIP]`.
  - Running fully in-core on GPU accelerators or in an hybrid CPU/GPU setup.
  - Using remeshed vortex methods (velocity-vorticity formulation).
  - And using a multiscale approach for scalars.

Numerical solver

- To be as efficient as possible, we focus on:
  - Cartesian grids.
  - Spectral method to recover the vorticity $\boldsymbol{\omega}$ from the velocity $\boldsymbol{u}$.
  - High order remeshing kernels for advection (and grid interpolation).
  - Explicit Runge-Kutta time integrators.
  - Centered finite differences for all other operators (stretching, ...).
  - Spectral method for diffusion and solenoidal projection.
- We use the HySoP library to perform operator and directional splitting



- To get more information about the solver, refer to part one.

Dimensionless velocity-vorticity formulation

- We rewrite our equations in their $(\boldsymbol{u}, \boldsymbol{\omega})$-formulation:

$$\rho = 1 + \alpha S + \gamma C \tag{3a}$$

$$\boldsymbol{\omega} = \nabla \wedge \boldsymbol{u} \tag{3b}$$

$$\nabla \cdot \boldsymbol{u} = 0 \tag{3c}$$

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{\omega} - (\boldsymbol{\omega} \cdot \nabla)\boldsymbol{u} = \Delta \boldsymbol{\omega} - \nabla \wedge (R_s S + C)\, \boldsymbol{e_z} \tag{3d}$$

$$\frac{\partial S}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, S = \frac{1}{S_c} \Delta S \tag{3e}$$

$$\frac{\partial C}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, C - V_p \frac{\partial C}{\partial z} = \frac{1}{\tau S_c} \Delta C \tag{3f}$$

- We perform operator splitting on eqs. (3d), (3e) and (3f).
- We further split those equations directionally.

Numerical results

2D results

Initial conditions

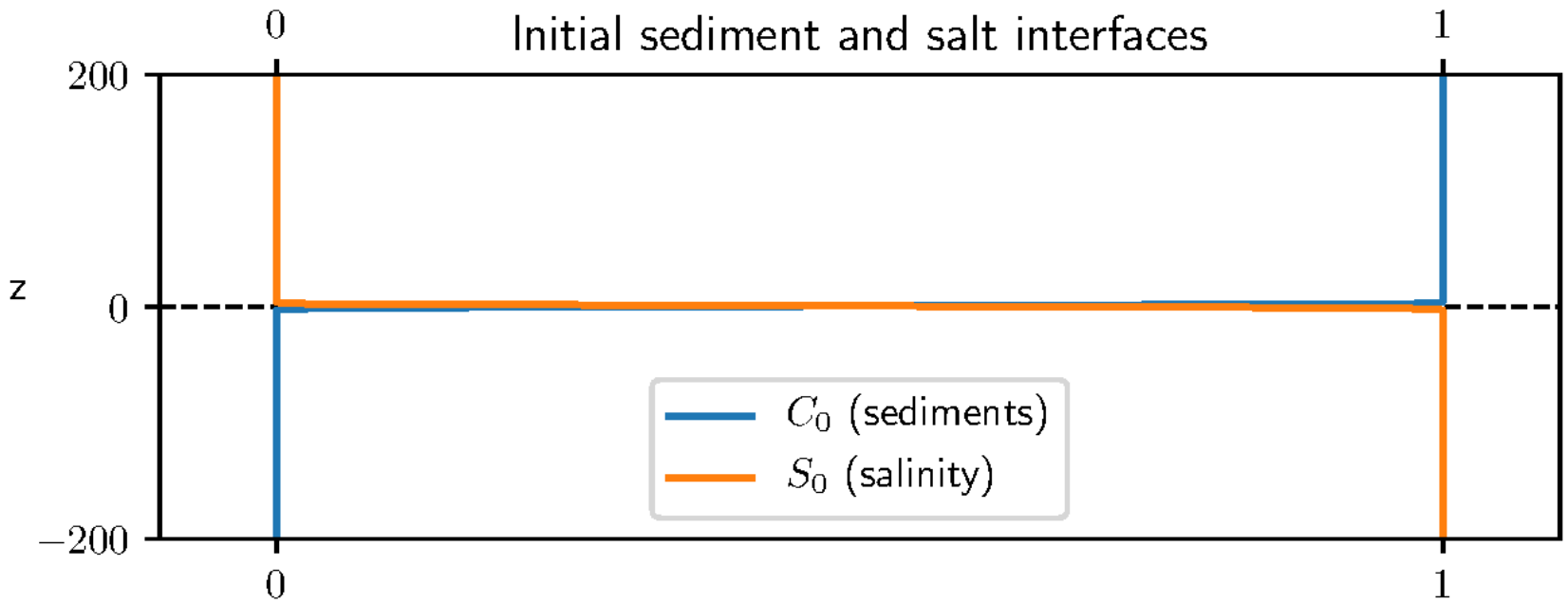
$$\boldsymbol{u}_0(x,z) = 0 \text{ and } \boldsymbol{\omega}_0(x,z) = 0$$

$$C_0(x,z) = \frac{1}{2}\left[1 + erf\left(\frac{z-\delta(x)}{l_0}\right)\right]$$

$$S_0(x,z) = 1 - C_0(x,z)$$

$$l_0 = 1.5, z \in [-600, +600]$$

$\delta(x)$ is a small initial random perturbation



Initial conditions

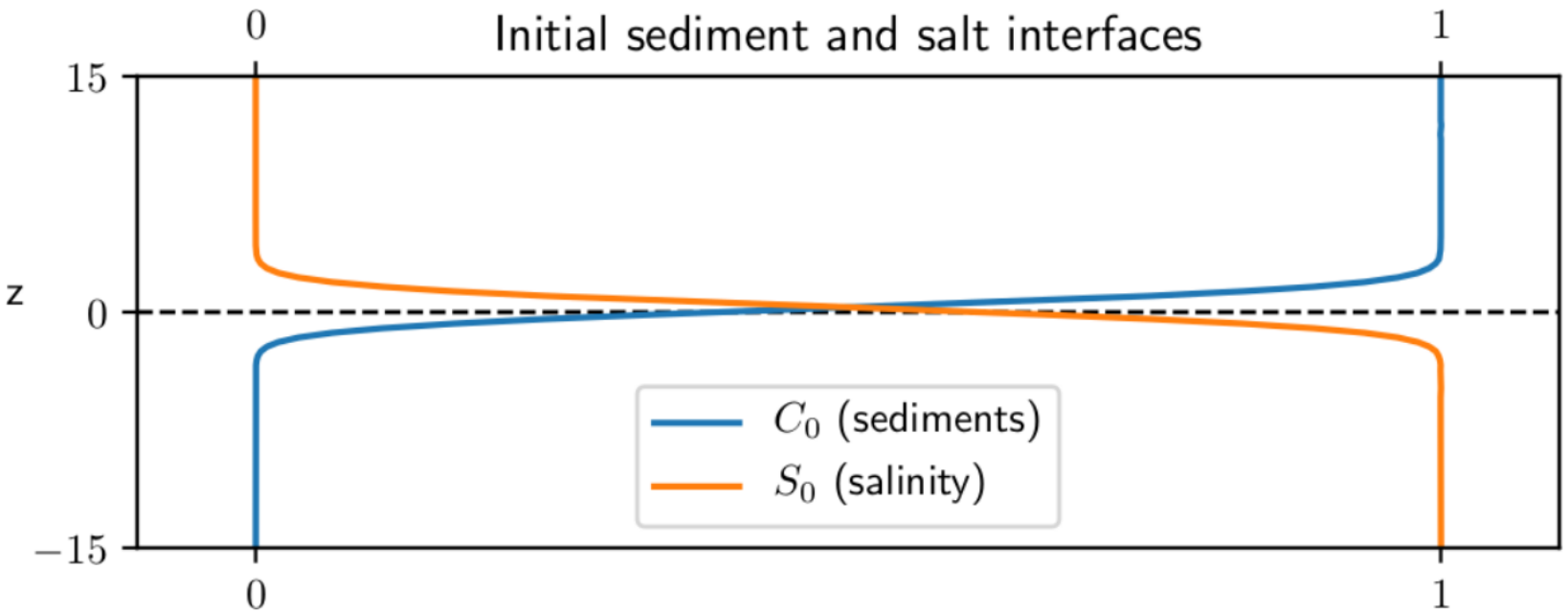
$$\boldsymbol{u}_0(x,z) = 0 \text{ and } \boldsymbol{\omega}_0(x,z) = 0$$

$$C_0(x,z) = \frac{1}{2}\left[1 + erf\left(\frac{z-\delta(x)}{l_0}\right)\right]$$

$$S_0(x,z) = 1 - C_0(x,z)$$

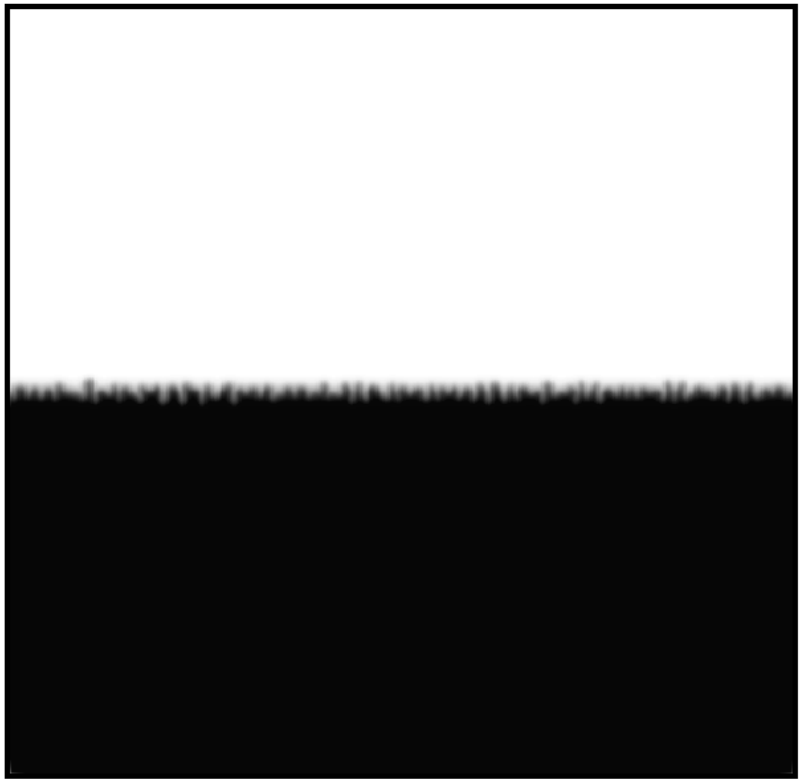
$$l_0 = 1.5, z \in [-600, +600]$$

$\delta(x)$ is a small initial random perturbation



2D results

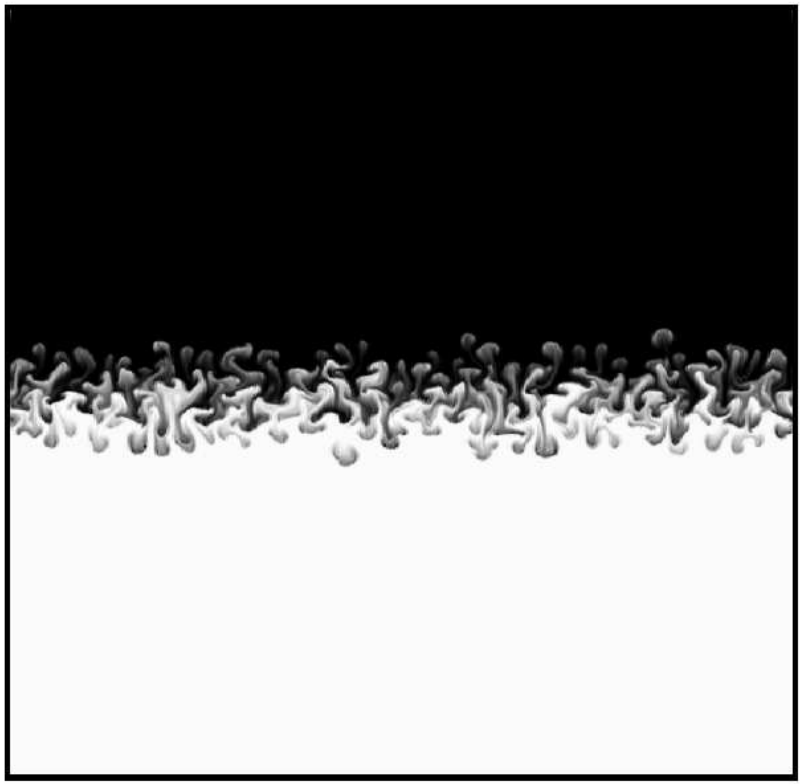
(a) C(t=40)



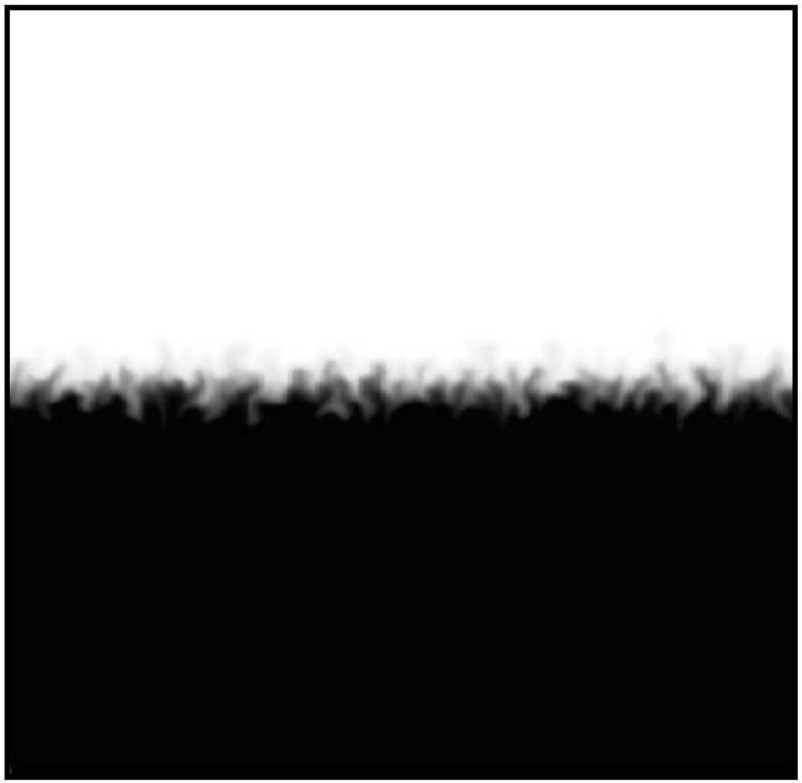
(b) S(t=40)

Snapshots of sediment and salinity concentration profiles

2D results



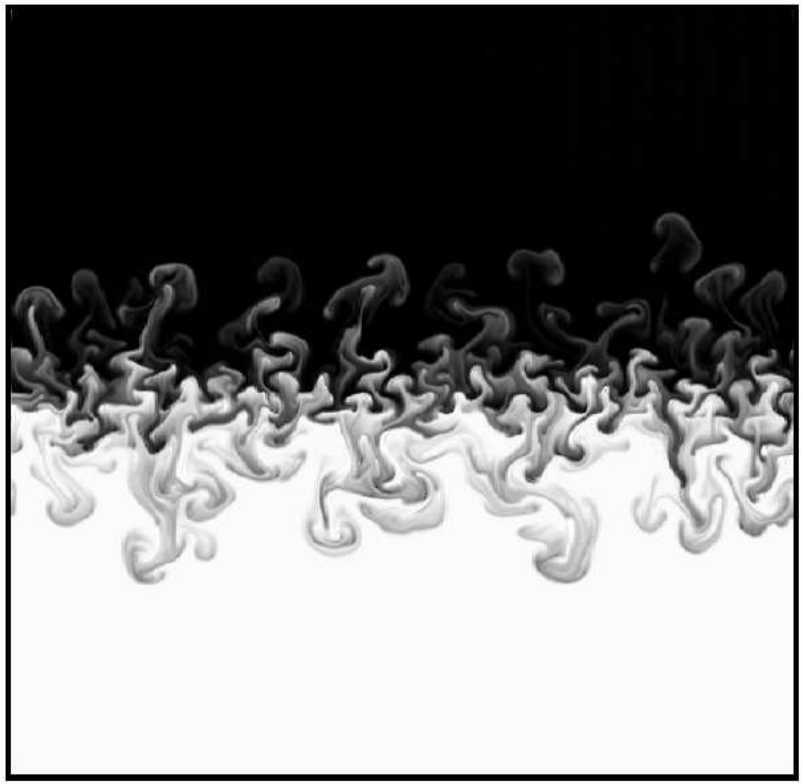
(c) C(t=100)



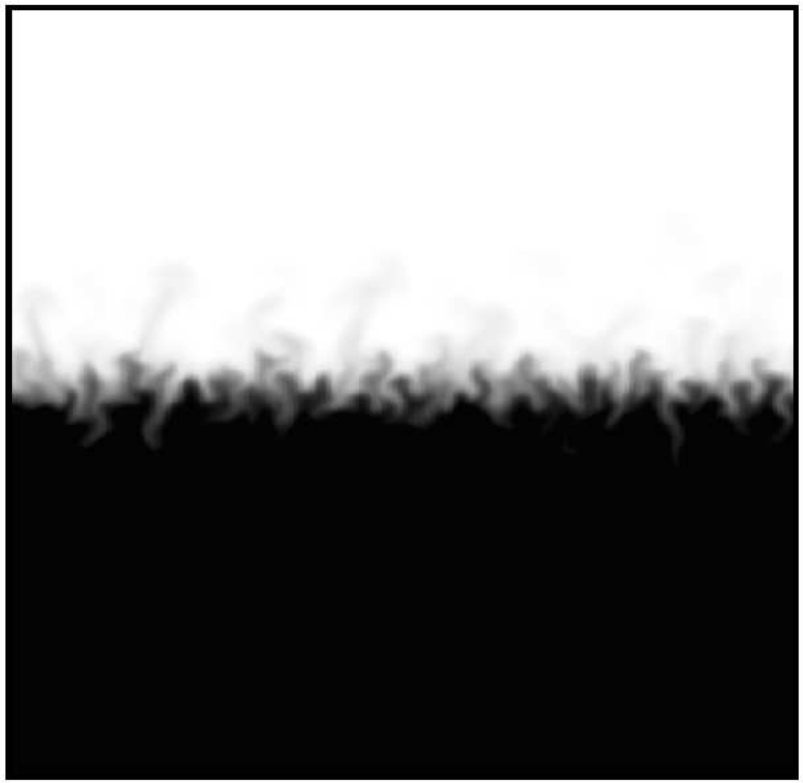
(d) S(t=100)

Snapshots of sediment and salinity concentration profiles

2D results



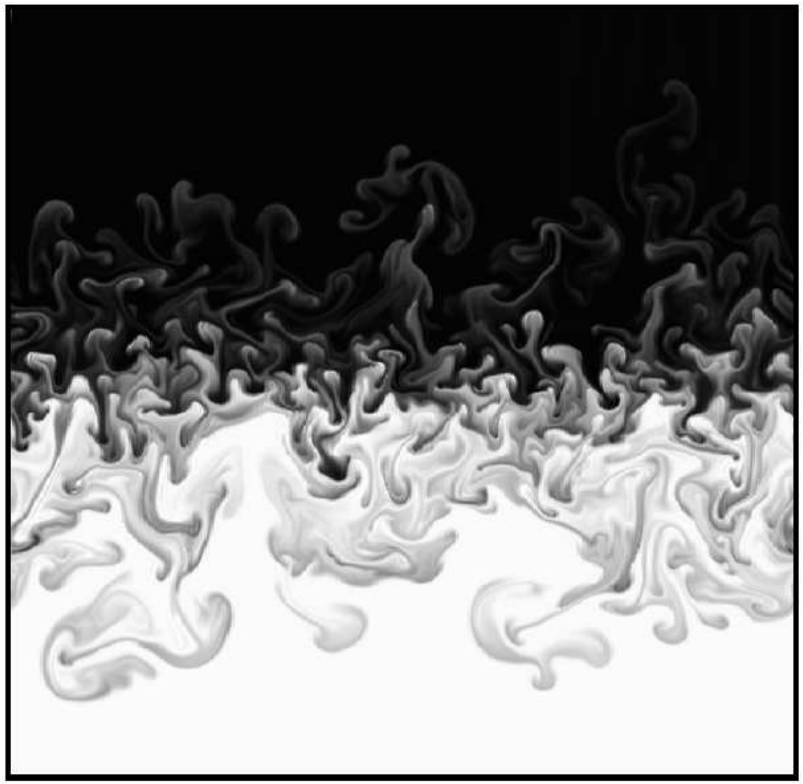
(e) C(t=200)



(f) S(t=200)

Snapshots of sediment and salinity concentration profiles

2D results

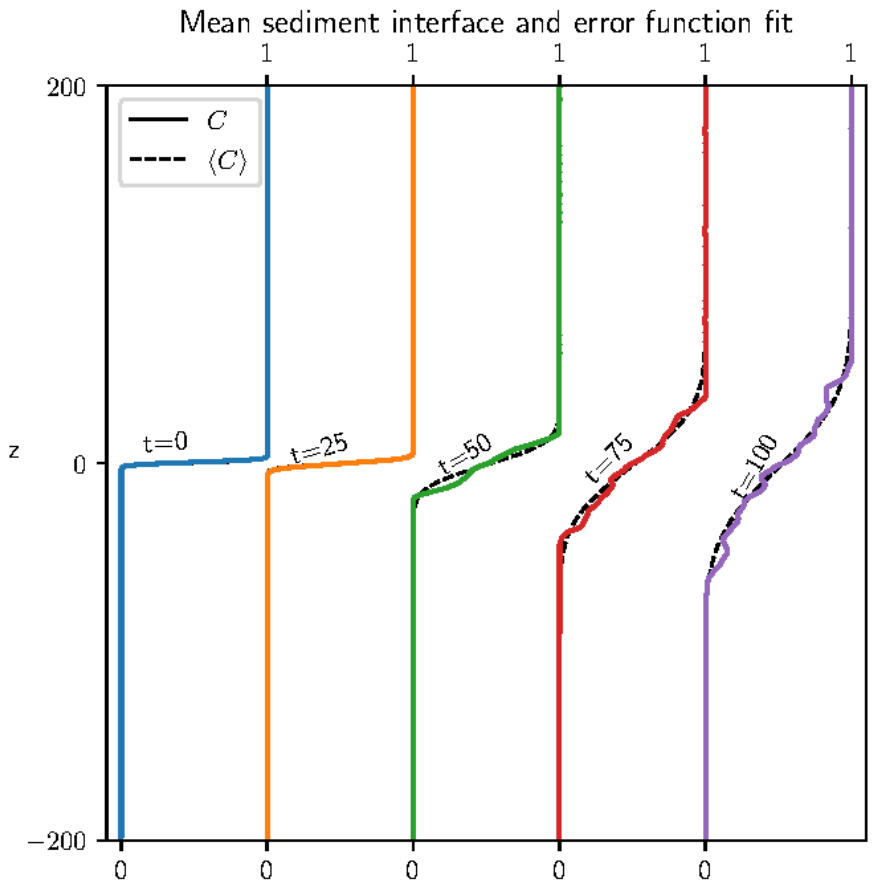


(g) C(t=300)

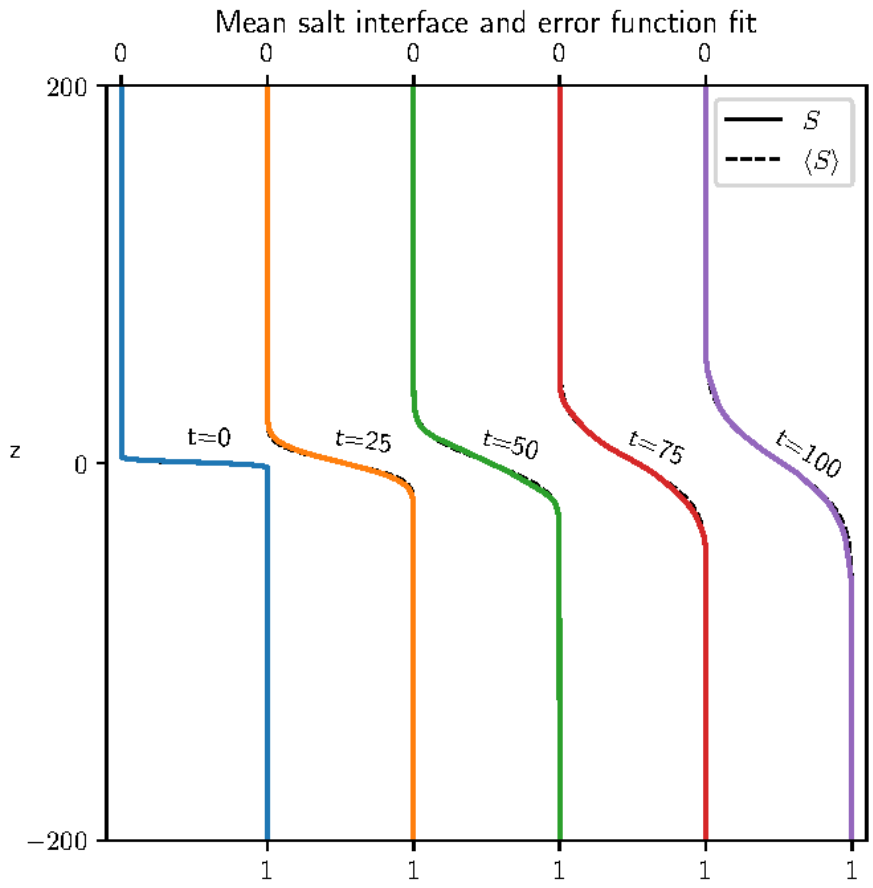
(h) S(t=300)

Snapshots of sediment and salinity concentration profiles

Horizontally averaged sediment and salt profiles



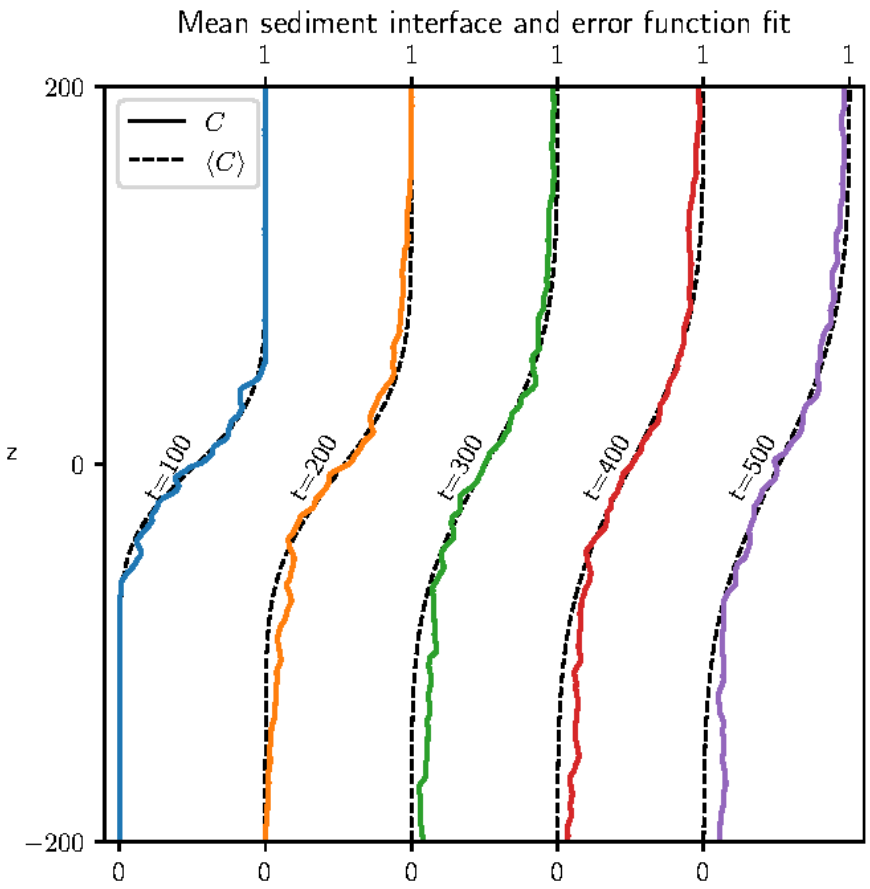
(i) Sediments



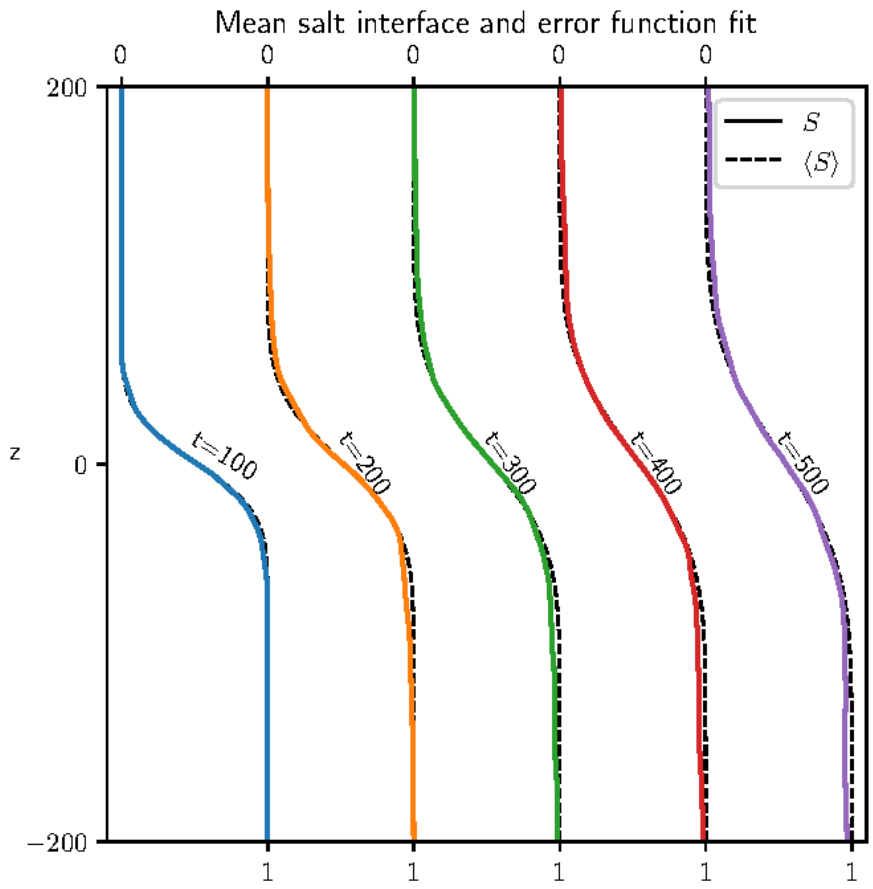
(j) Salinity

Horizontally averaged sediment and salinity concentration profiles (dotted lines represent error function fit).

Horizontally averaged sediment and salt profiles



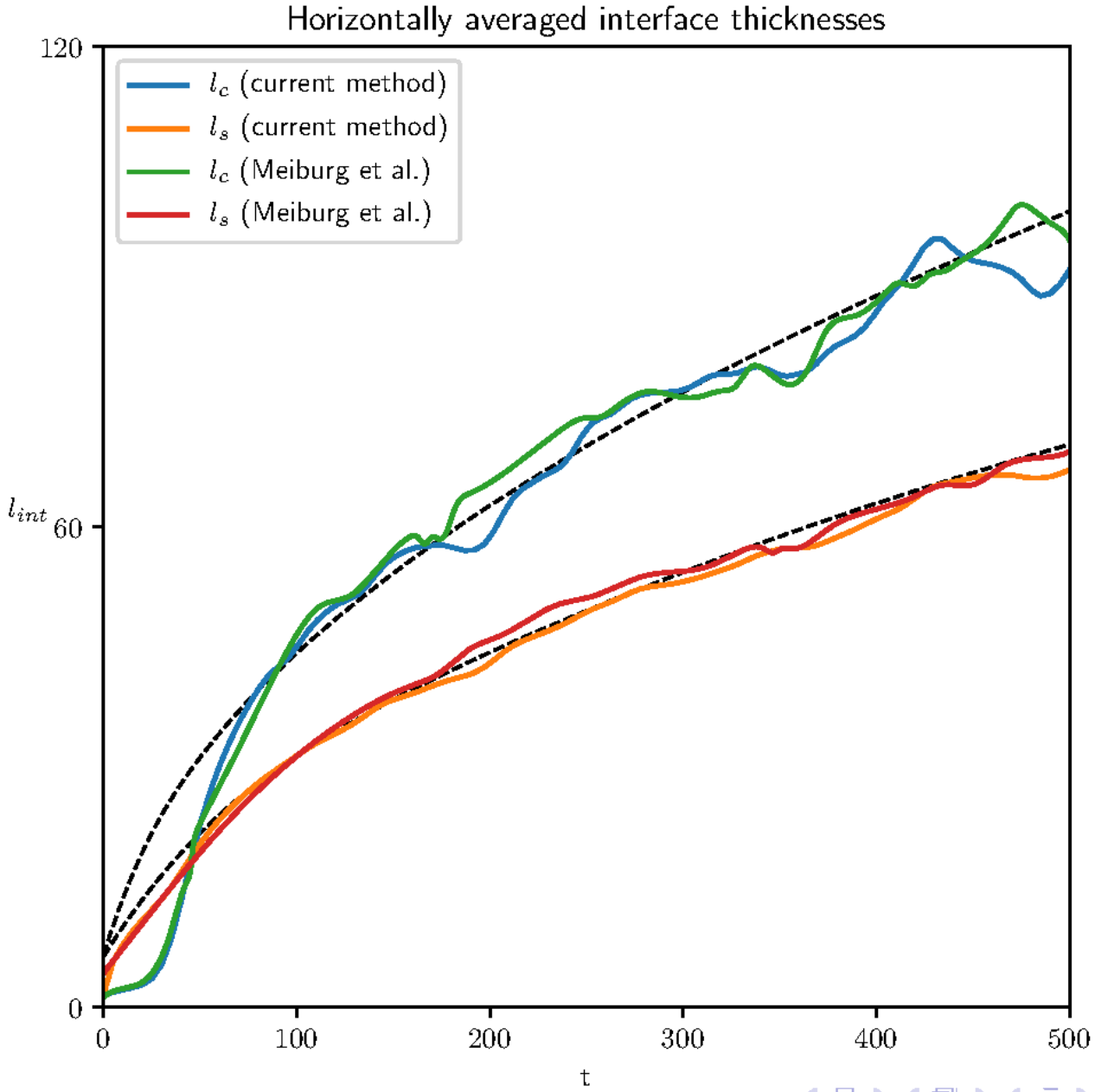
(k) Sediments



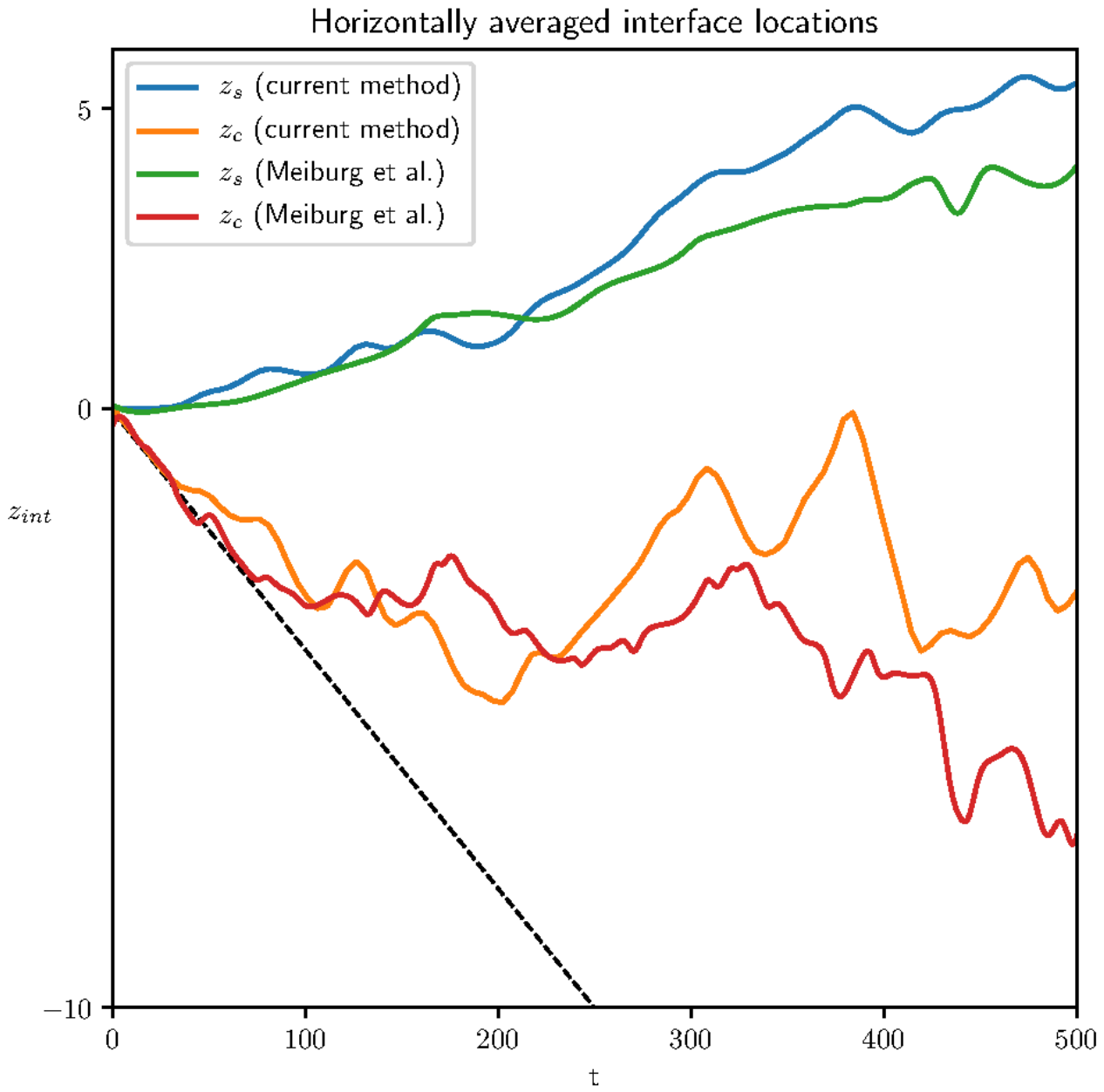
(l) Salinity

Horizontally averaged sediment and salinity concentration profiles (dotted lines represent error function fit).

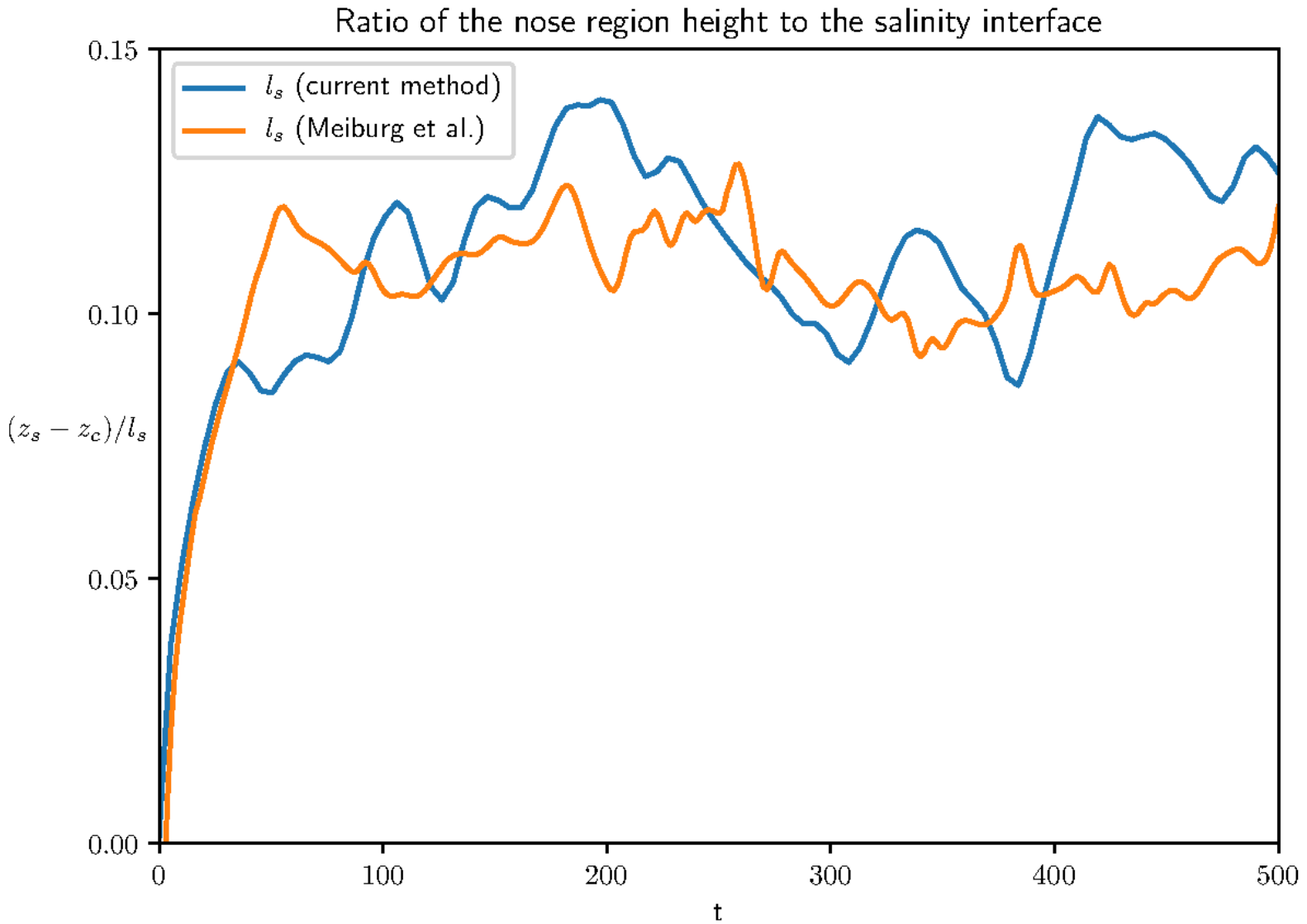
Horizontally averaged interface thicknesses



Horizontally averaged interface locations



Nose region ratio - comparison with [2]



2D CPU vs GPU performances (without I/O)

Numerical setup:

- $\Omega = [0, 750] \times [-600, +600], t \in [0, 500]$
- $S_c = 0.7, \tau = 25, V_p = 0.04, R_s = 2$

Software:

- We use the `HySoP` library (FDC2, RK4, L4_2, single precision).
- Same `OpenCL` backend for both `CPU` and `GPU`.

Hardware:

- **CPU:** `2 x Intel Xeon E5-2695V4 CPU (18 cores each)`
- **GPU:** `Nvidia GeForce GTX 980 Ti`

Simulation Time		iteration (ms)		total (s)	
Resolution	Iterations	CPU	GPU	CPU	GPU
512x2048	7949	98.7	26.8	784.9	213.2
1024x4096	18763	228.3	31.9	4283.3	598.9
2048x8192	66590	728.7	84.7	48523.7	5637.4

3D results

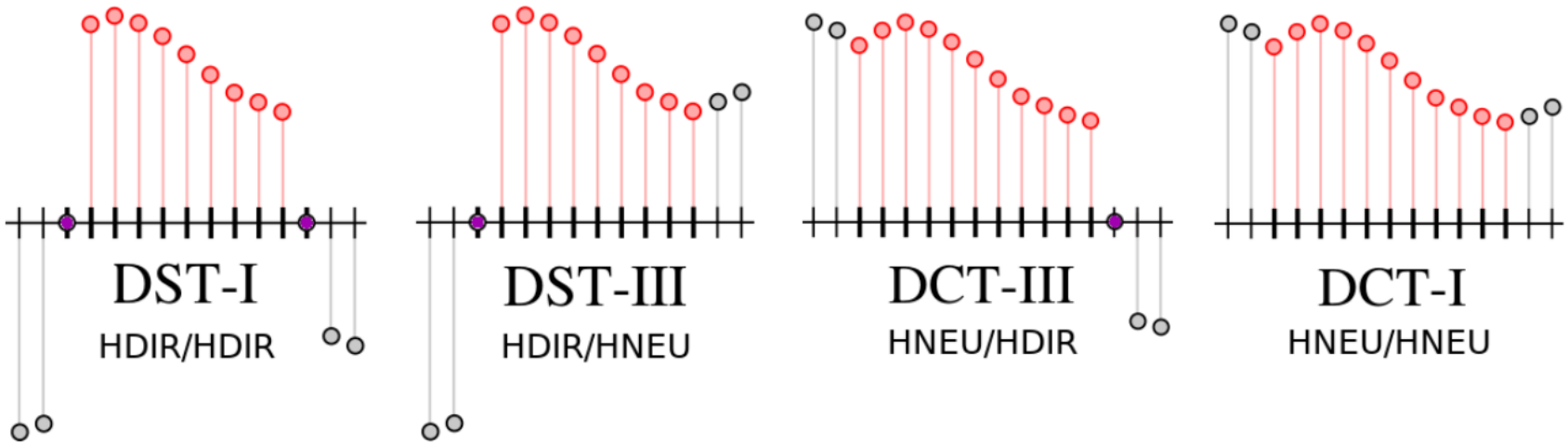
3D simulation

Numerical setup:

- $\Omega = [-110, +65] \times [0, 100]^2, t \in [0, 150]$
- $S_c = 7, \tau = 25, V_p = 0.04, R_s = 2$
- $N = N_x \times N_y \times N_z = 1537 \times 512 \times 512$
- 1.5GB per scalar field (cannot run on a single GPU)

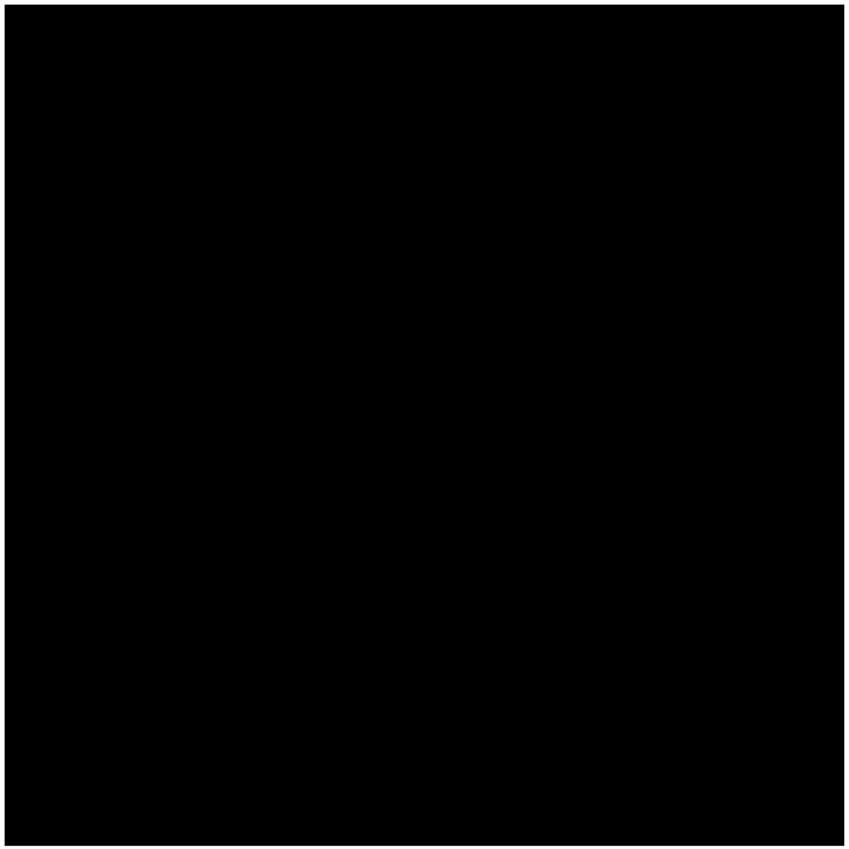
We change some operators:

- The 2D solver used a periodic solver with penalization.
- Now we use a homogeneous Dirichlet or Neumann spectral solver:

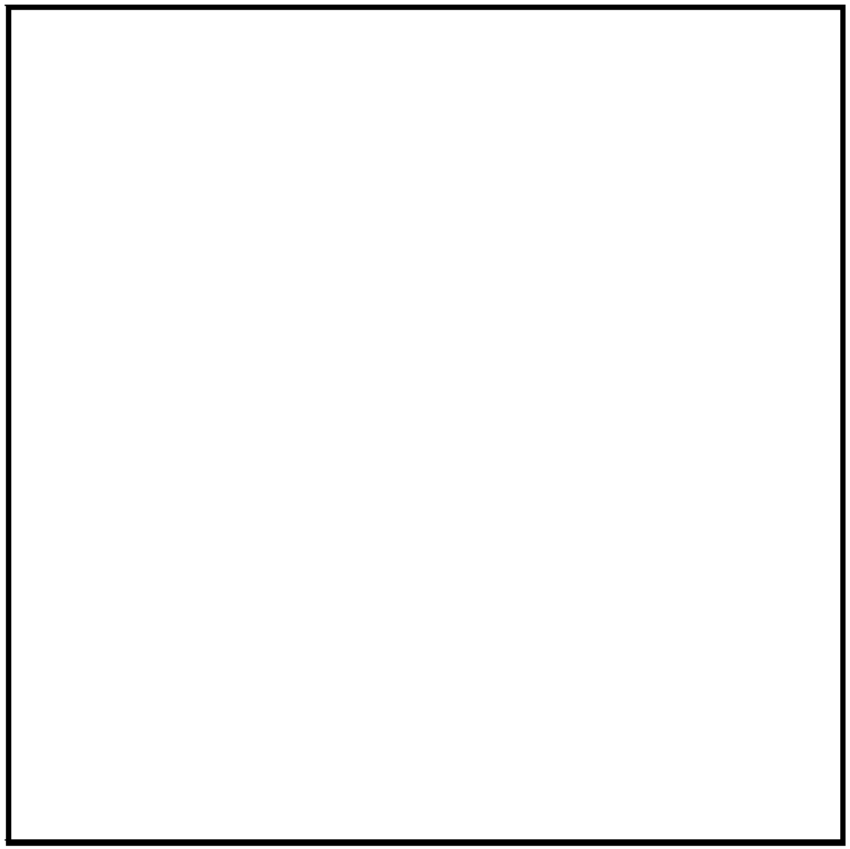


- We also use a spectral implicit solver for the diffusion.

3D Results - Horizontal 2D slices



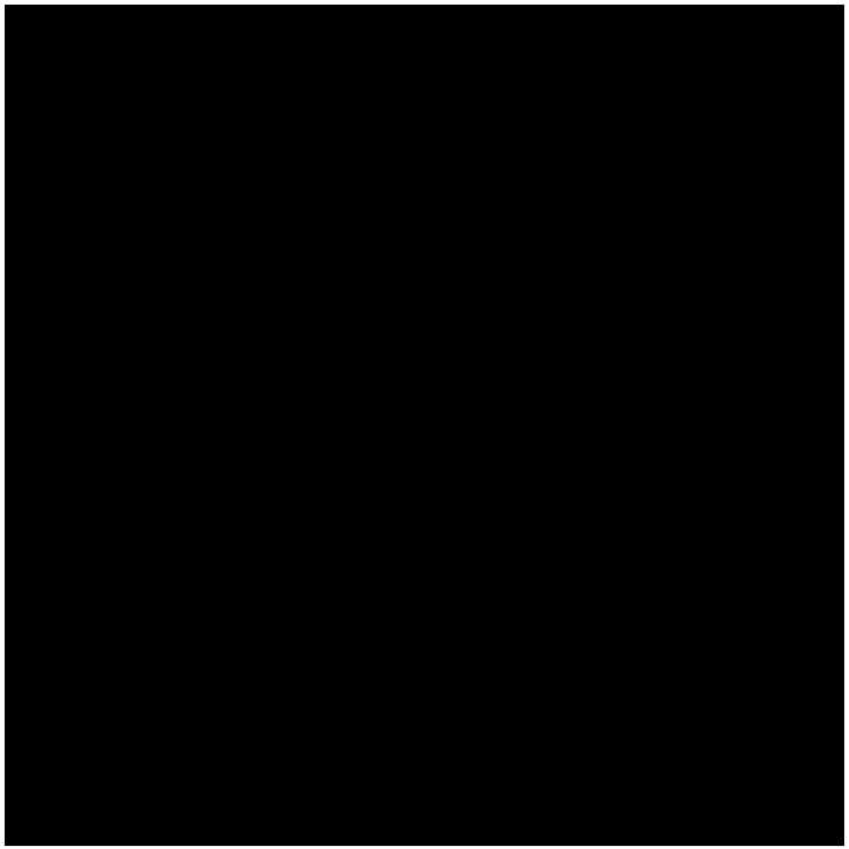
(m) C(t=50, z=+10)



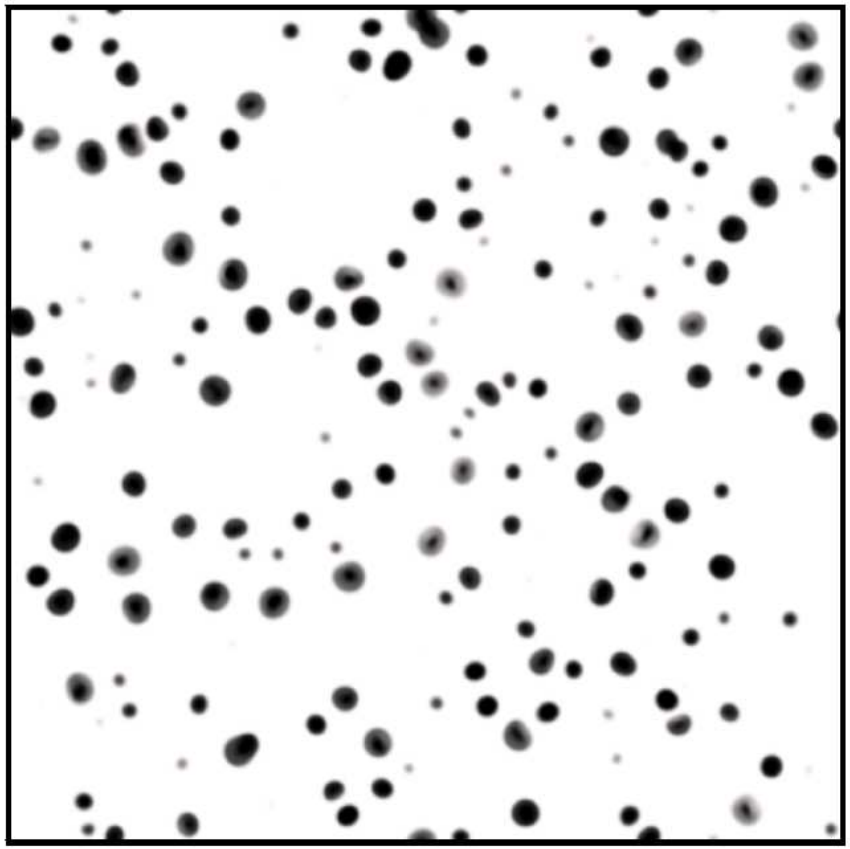
(n) C(t=50, z=-10)

Horizontal sediment profiles $C(z)$ at $z = \pm 10$.

3D Results - Horizontal 2D slices



(o) C(t=75, z=+10)

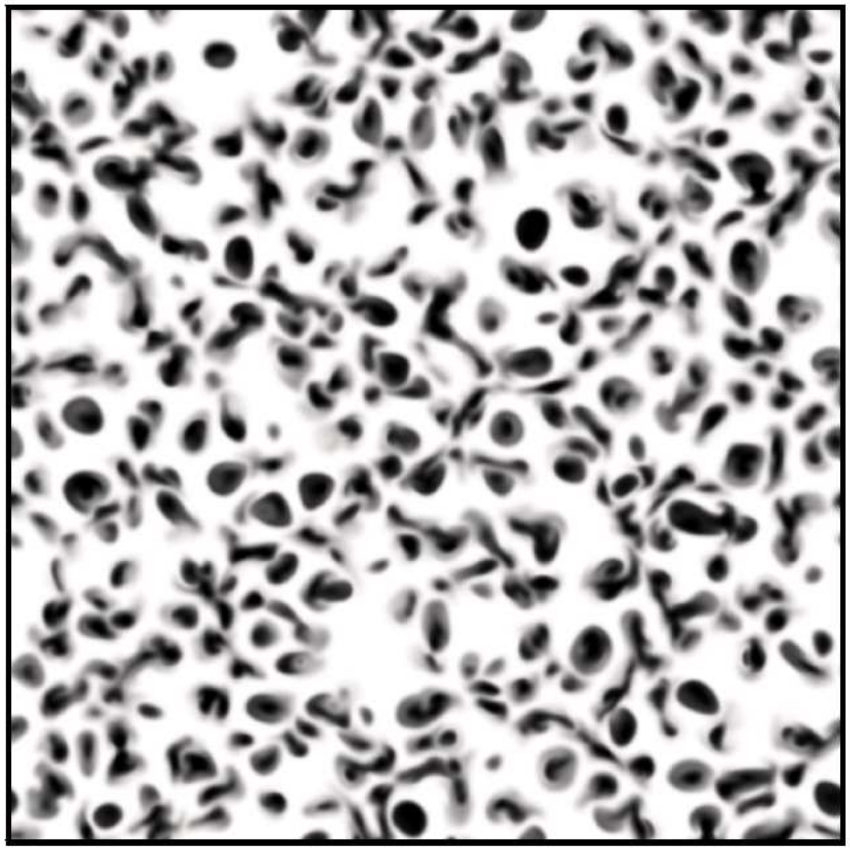


(p) C(t=75, z=-10)

Horizontal sediment profiles $C(z)$ at $z = \pm 10$.

3D Results - Horizontal 2D slices

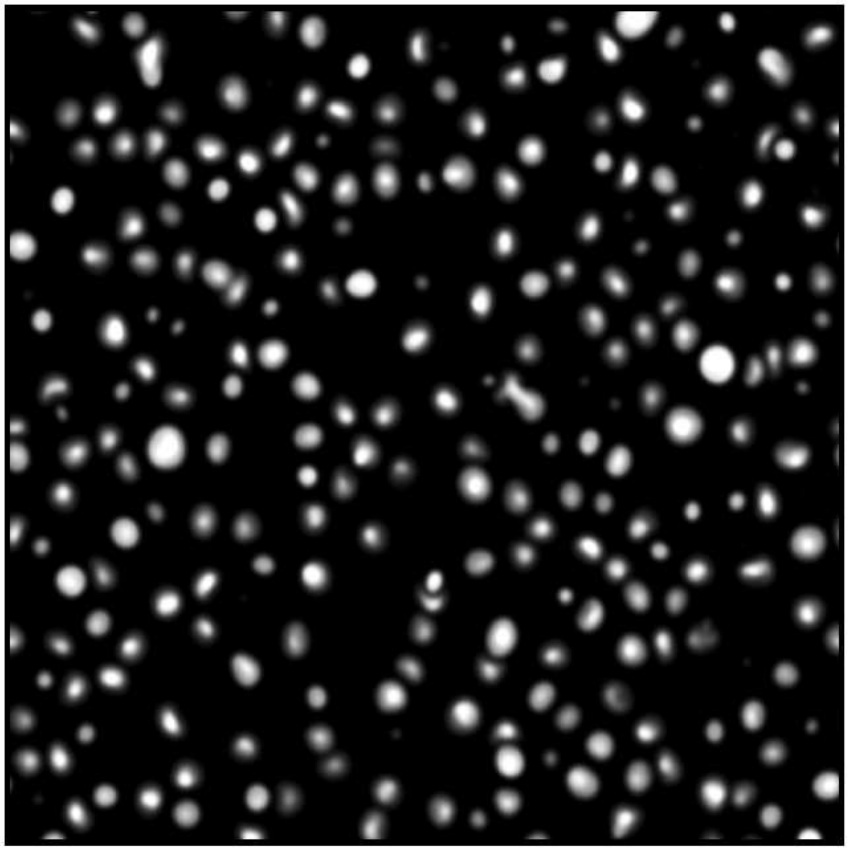
(q) C(t=100, z=+10)



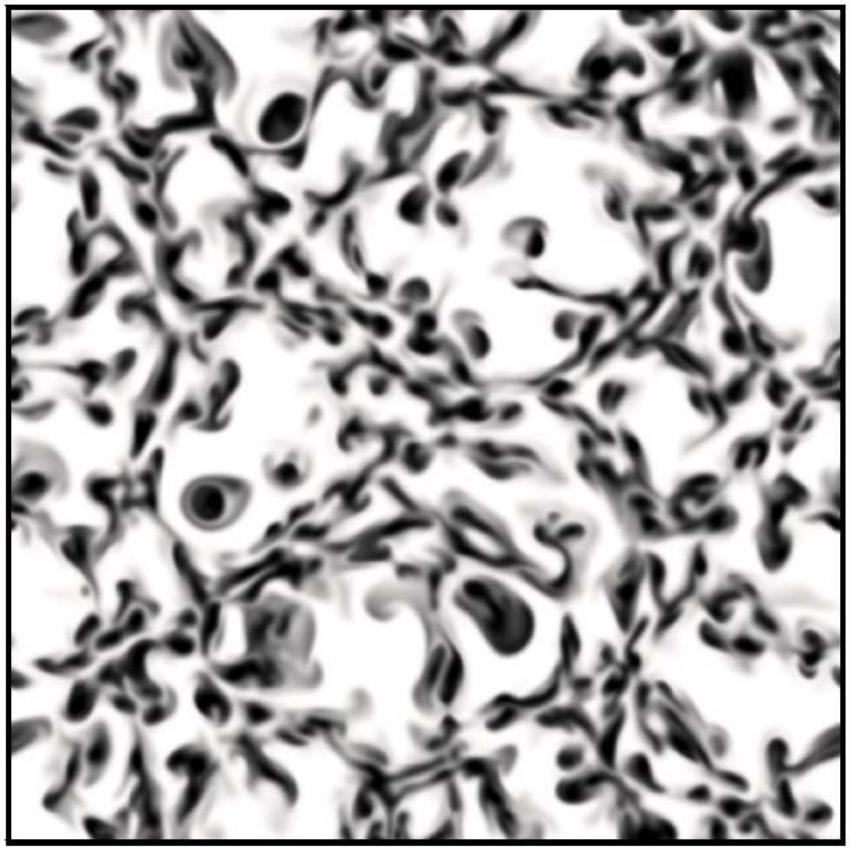
(r) C(t=100, z=-10)

Horizontal sediment profiles $C(z)$ at $z = \pm 10$.

3D Results - Horizontal 2D slices



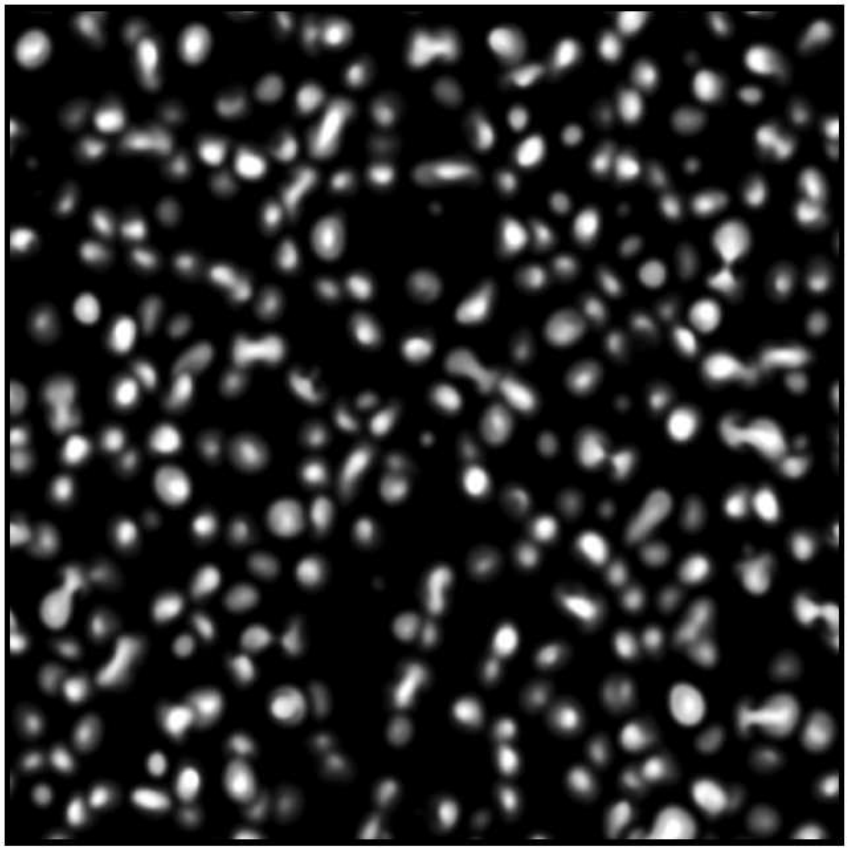
(s) C(t=125, z=+10)



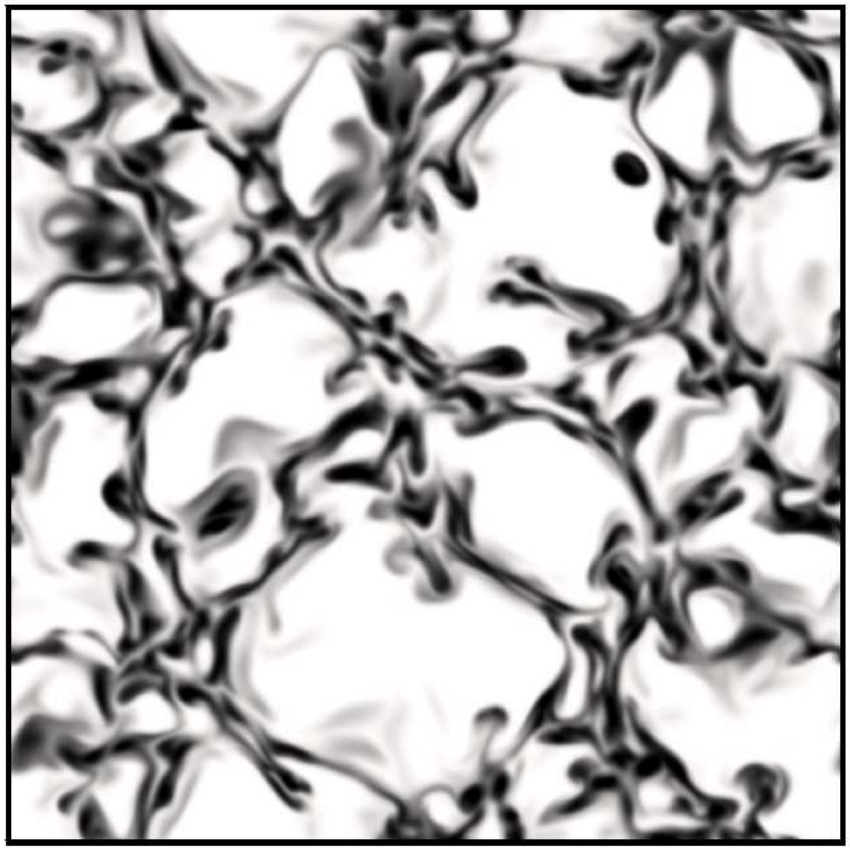
(t) C(t=125, z=-10)

Horizontal sediment profiles $C(z)$ at $z = \pm 10$.

3D Results - Horizontal 2D slices



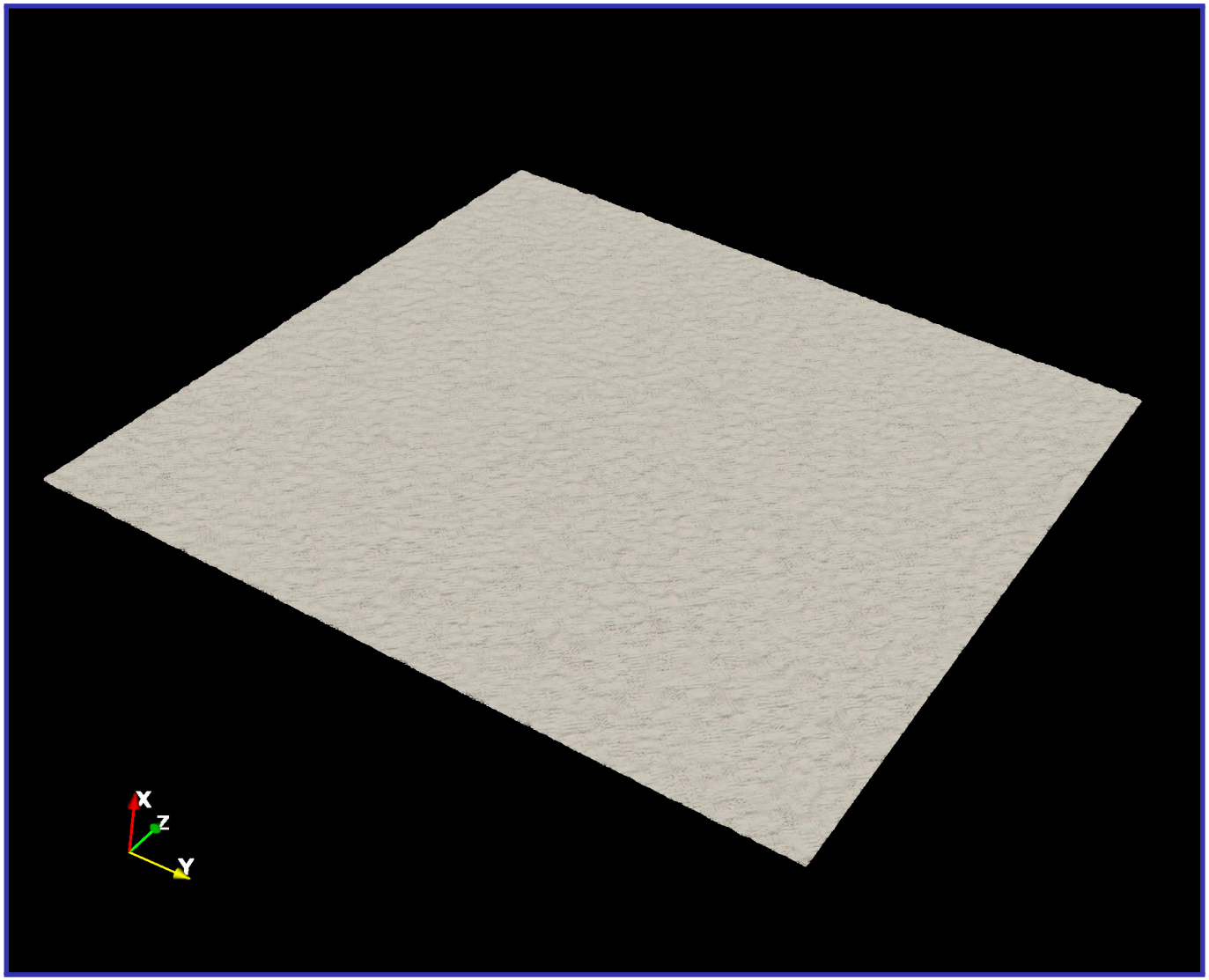
(u) C(t=150, z=+10)



(v) C(t=150, z=-10)

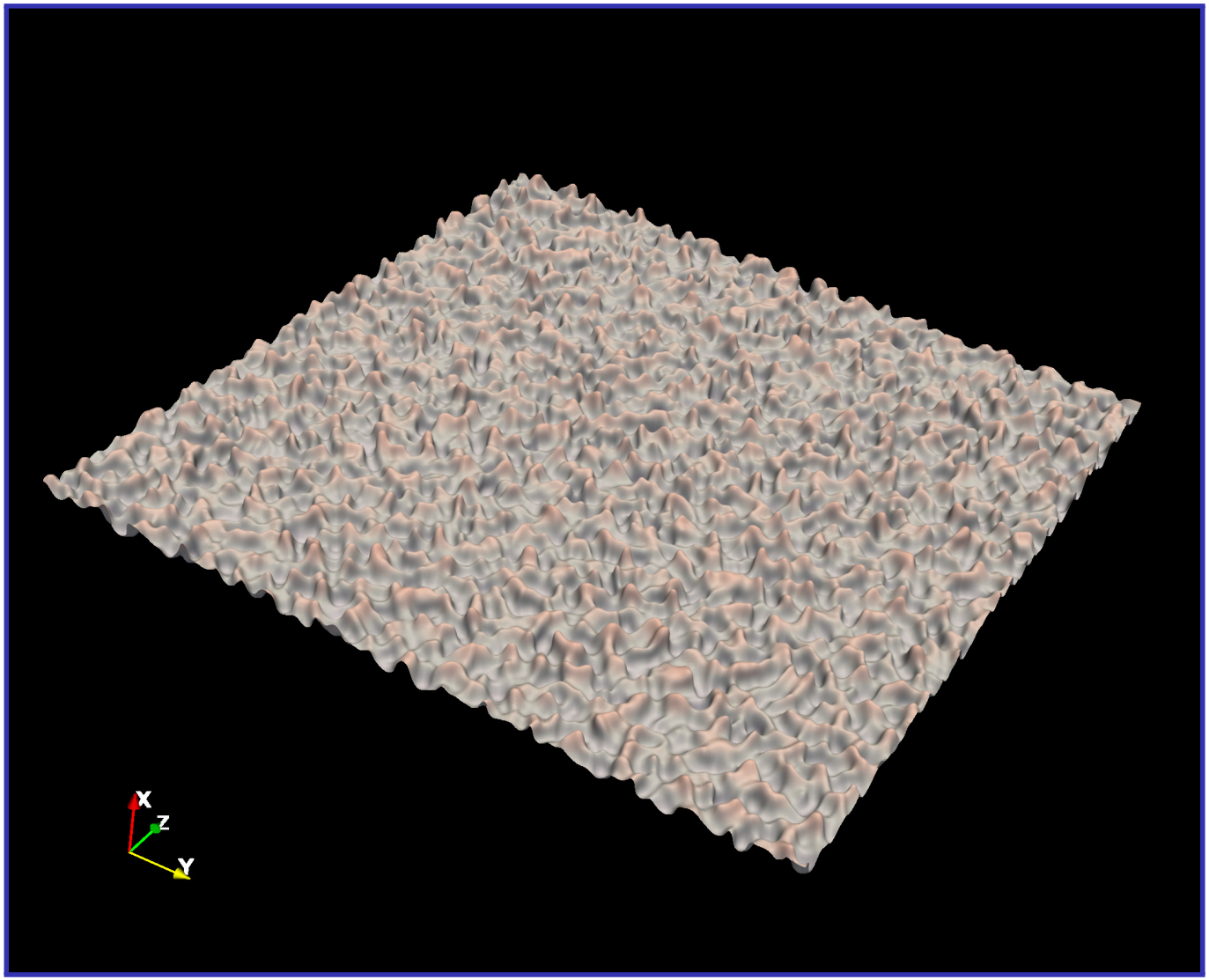
Horizontal sediment profiles $C(z)$ at $z = \pm 10$.

3D Results - levelset at C=0.5 (upper view, positive z)



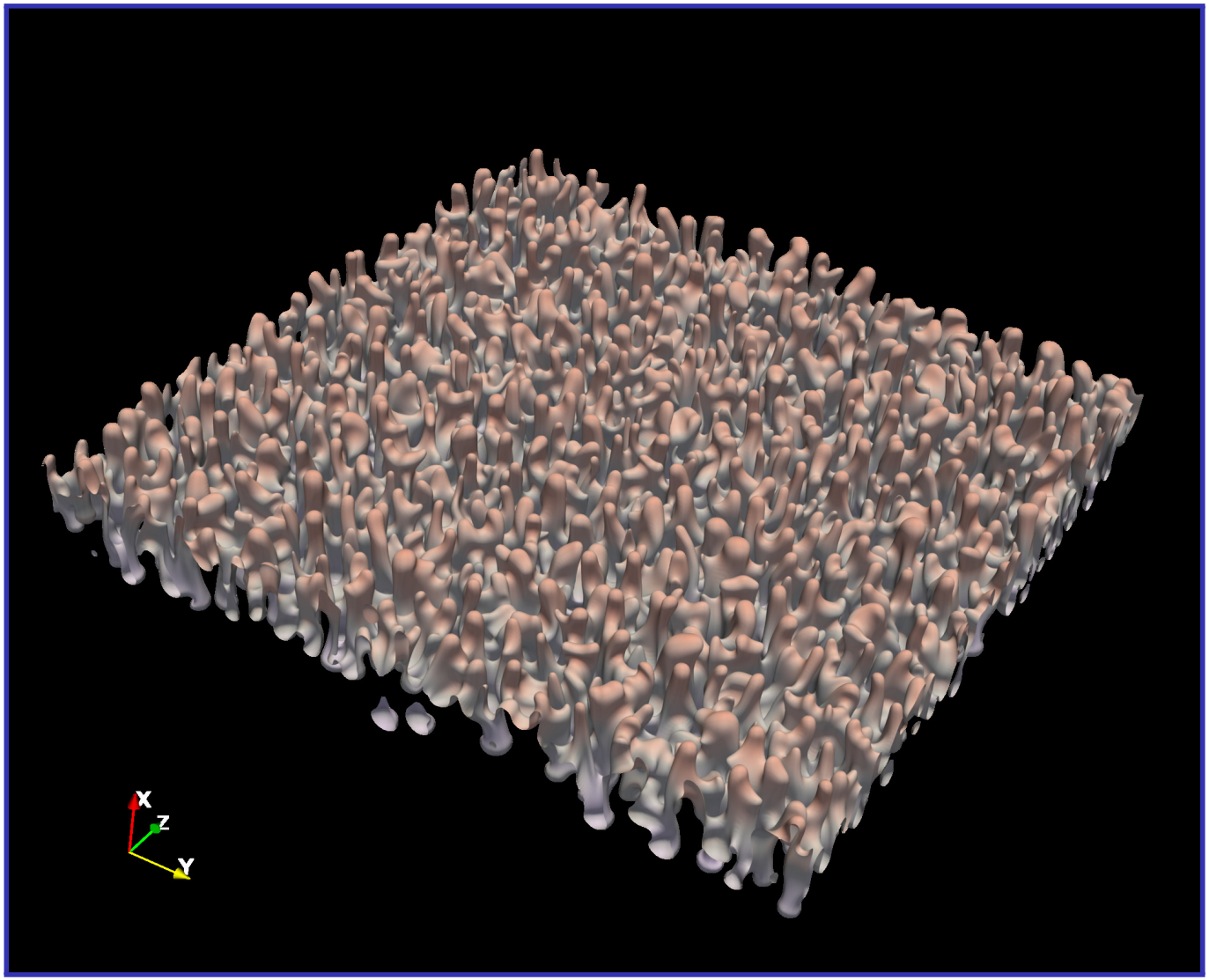
Sediment levelset $C(x, y, z) = 0.5$ at $t = 25$

3D Results - levelset at C=0.5 (upper view, positive z)



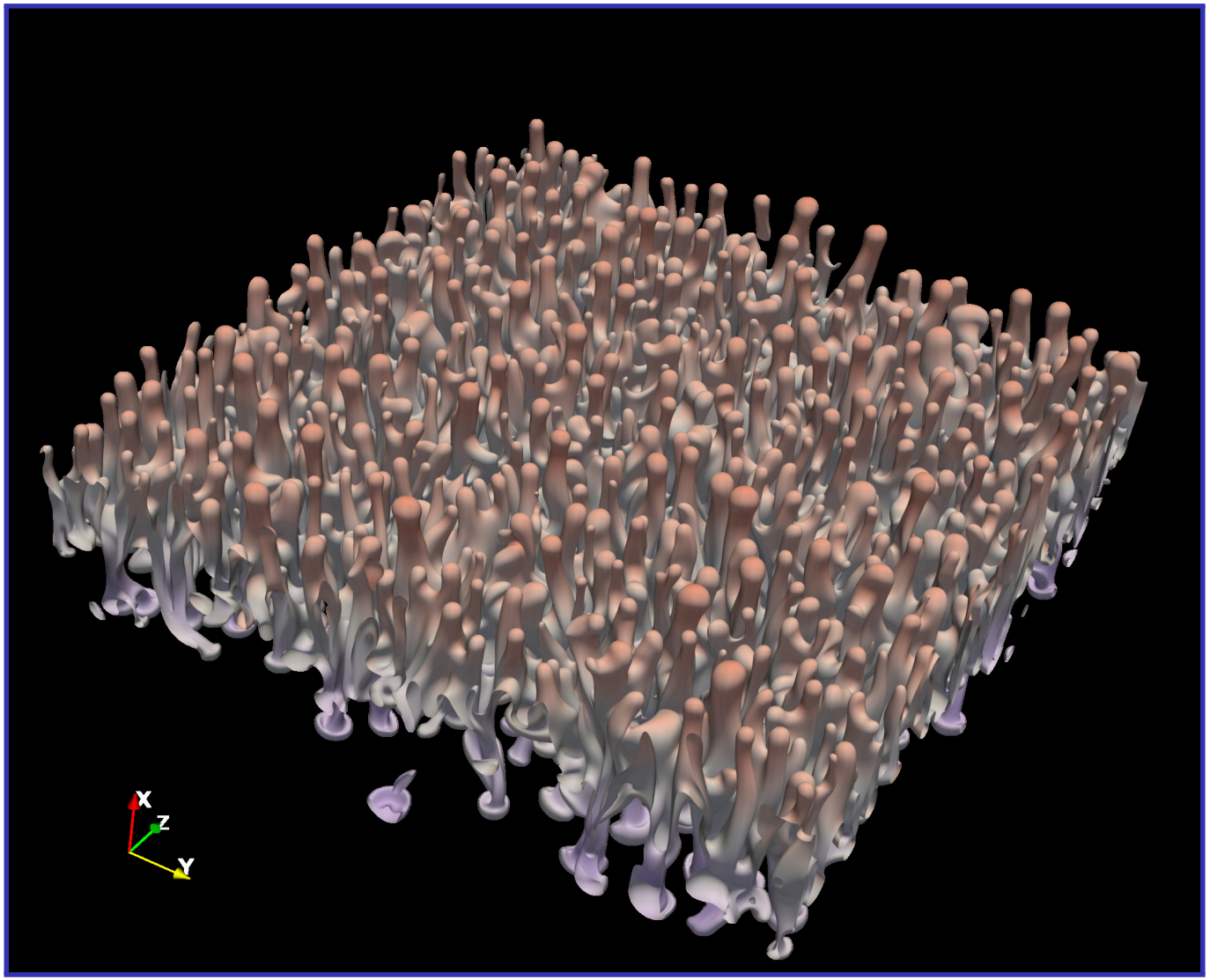
Sediment levelset $C(x, y, z) = 0.5$ at $t = 50$

3D Results - levelset at C=0.5 (upper view, positive z)



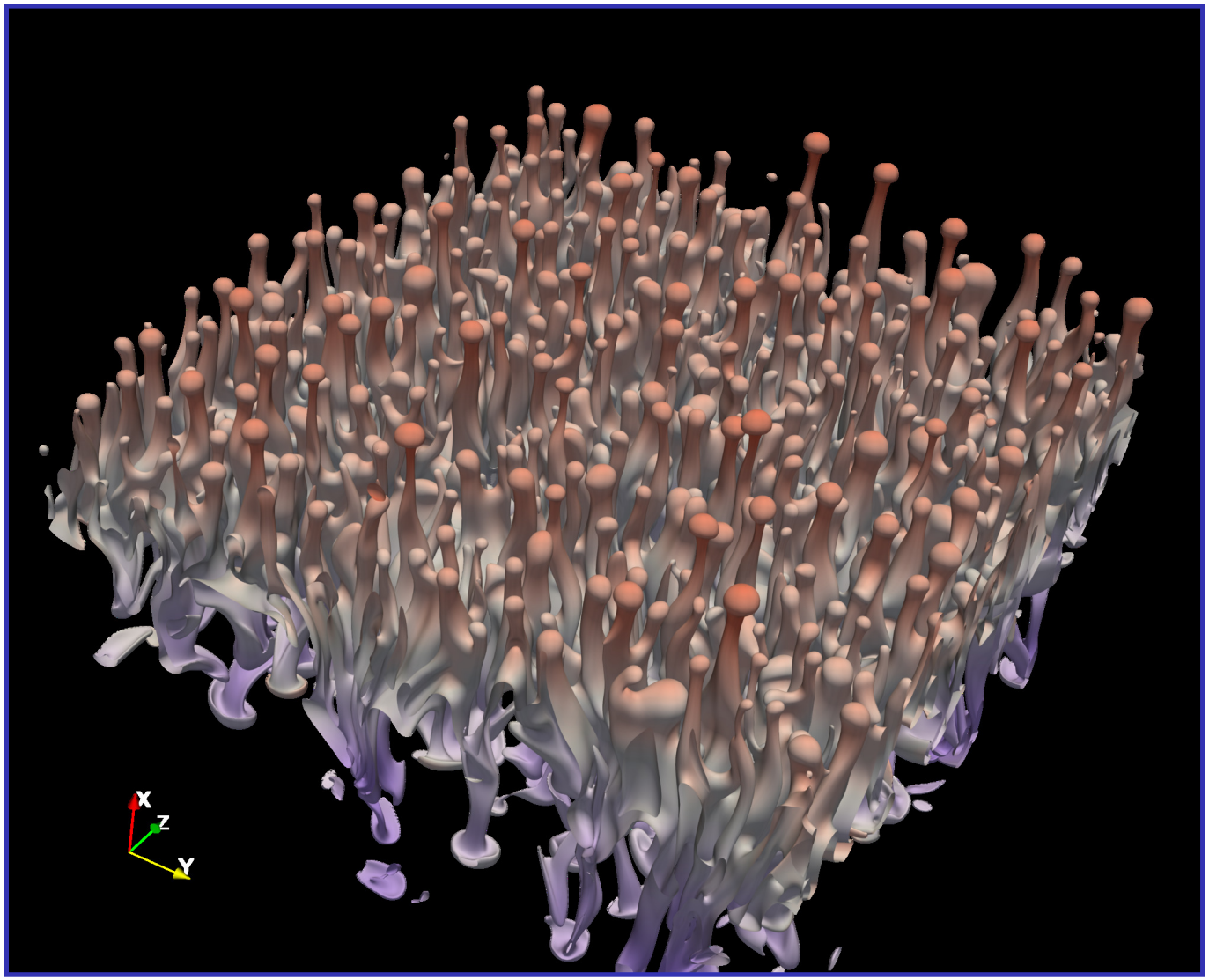
Sediment levelset $C(x, y, z) = 0.5$ at $t = 75$

3D Results - levelset at C=0.5 (upper view, positive z)



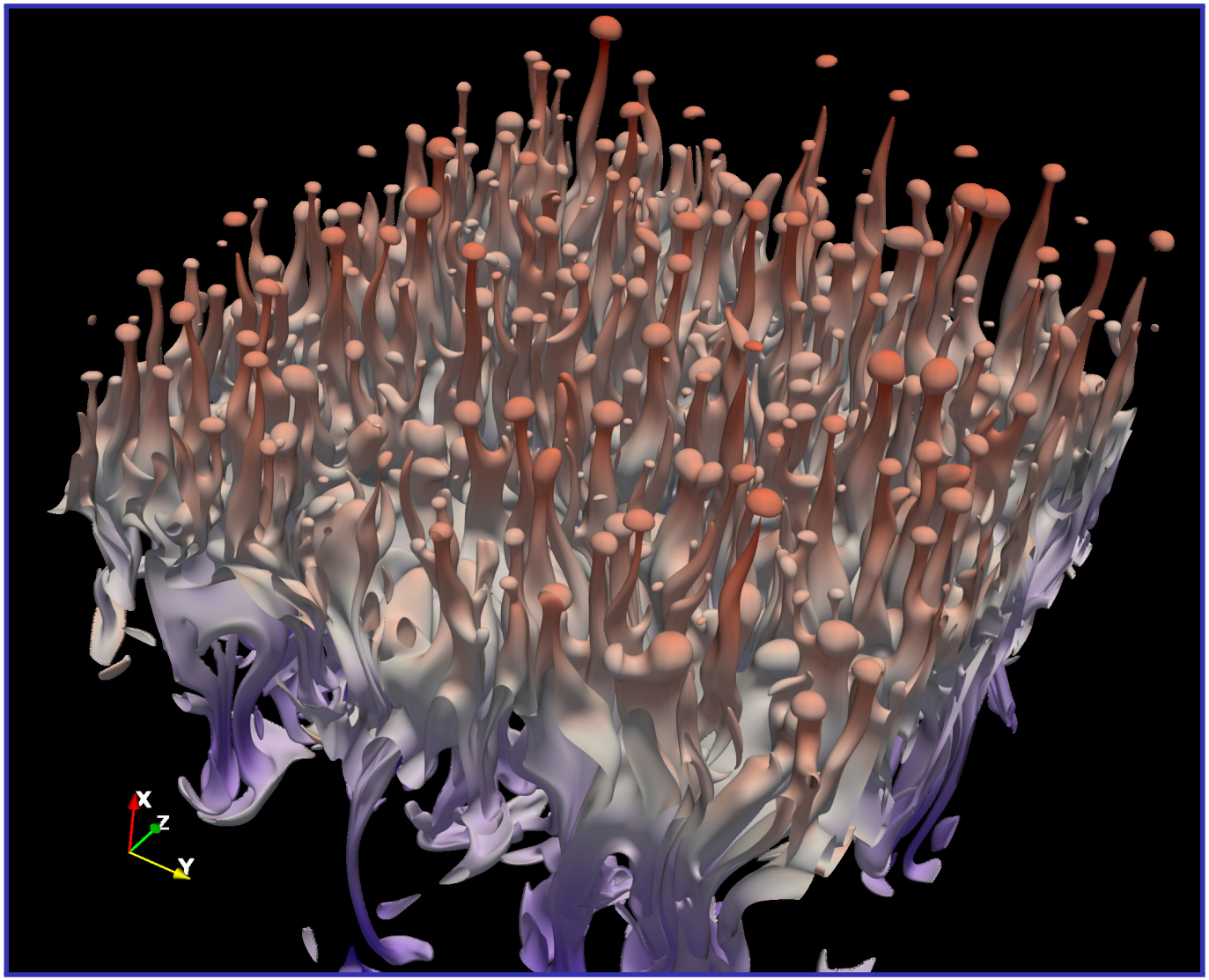
Sediment levelset $C(x, y, z) = 0.5$ at $t = 100$

3D Results - levelset at C=0.5 (upper view, positive z)



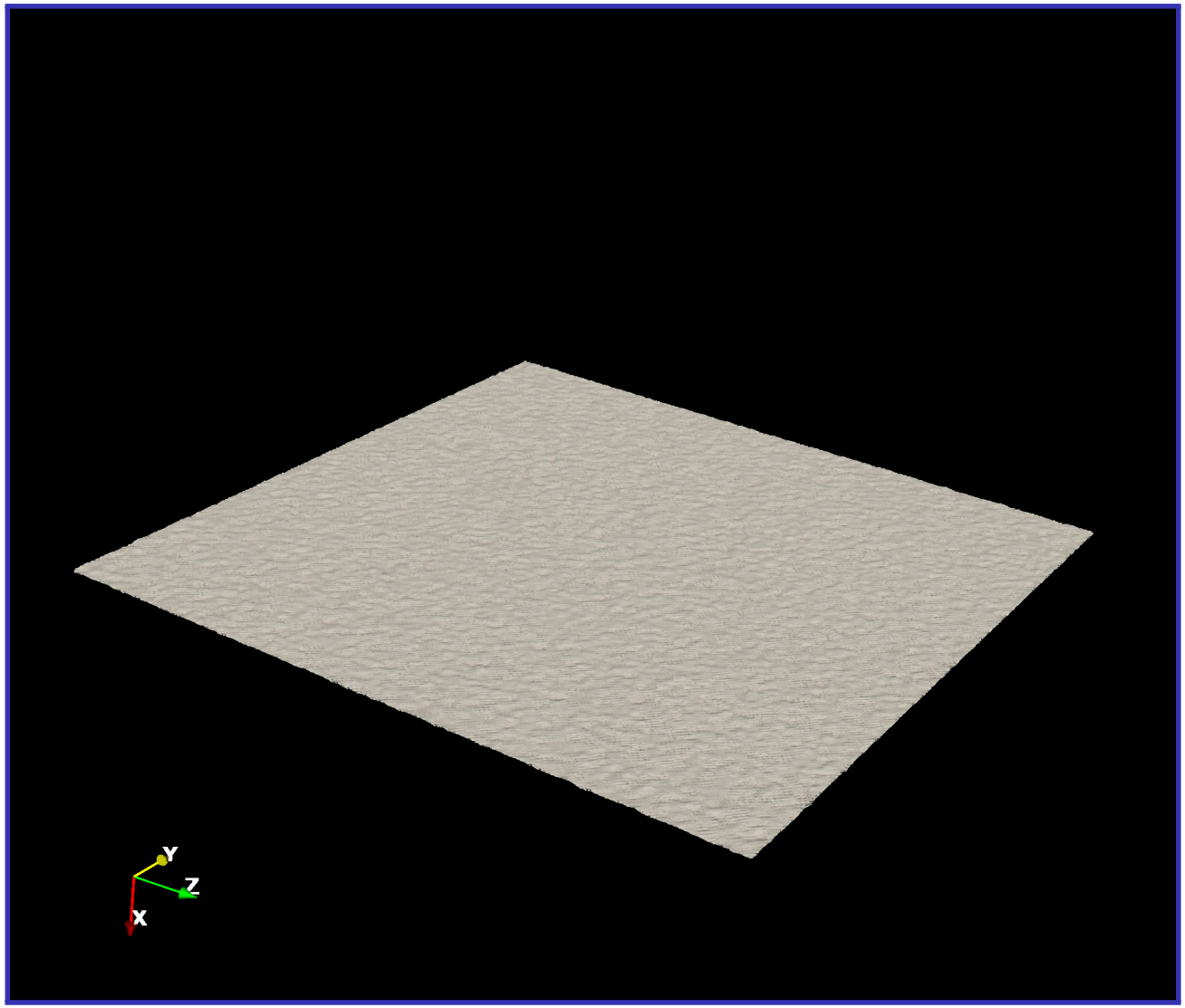
Sediment levelset $C(x, y, z) = 0.5$ at $t = 125$

3D Results - levelset at C=0.5 (upper view, positive z)



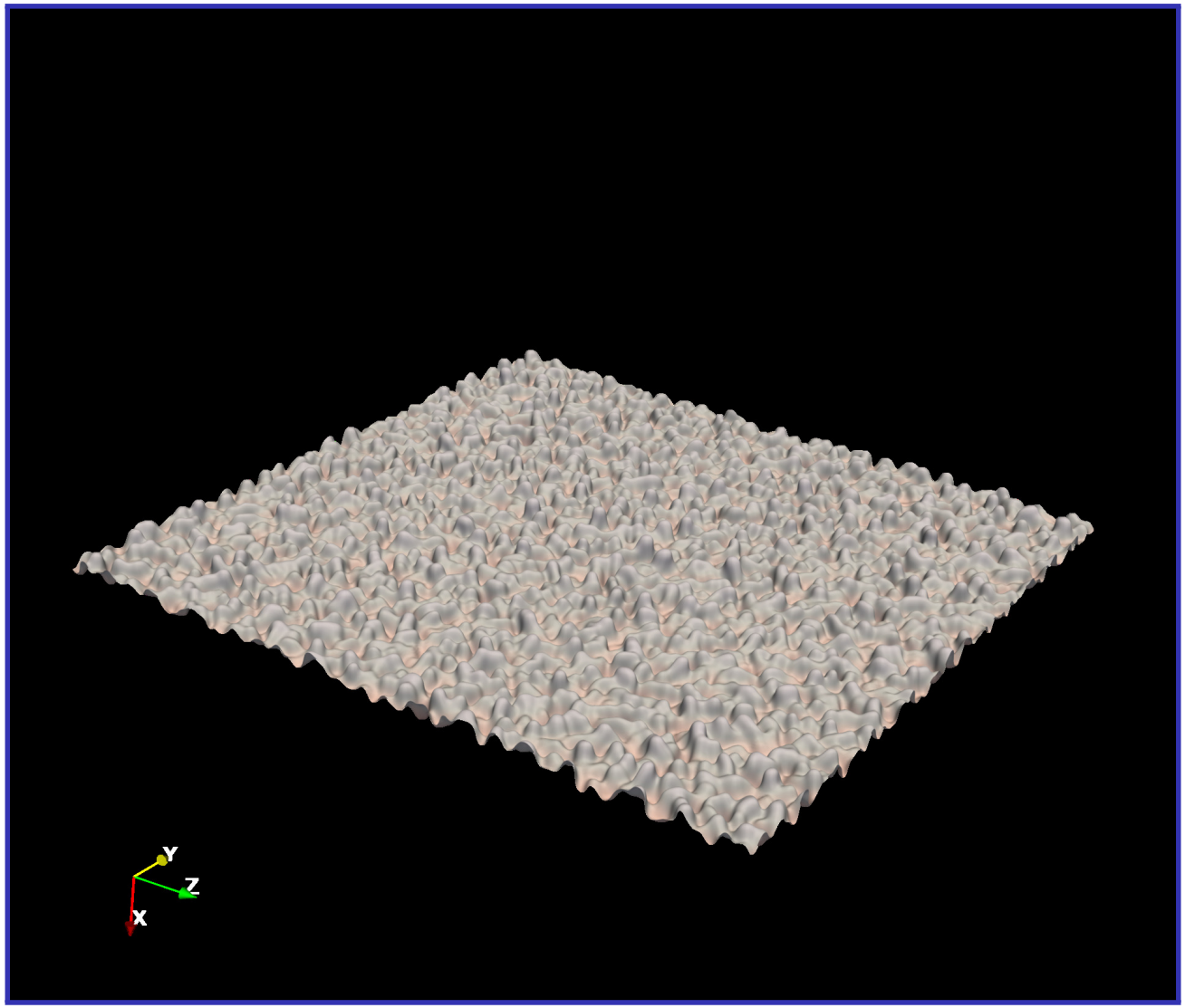
Sediment levelset $C(x, y, z) = 0.5$ at $t = 150$

3D Results - levelset at C=0.5 (lower view, negative z)



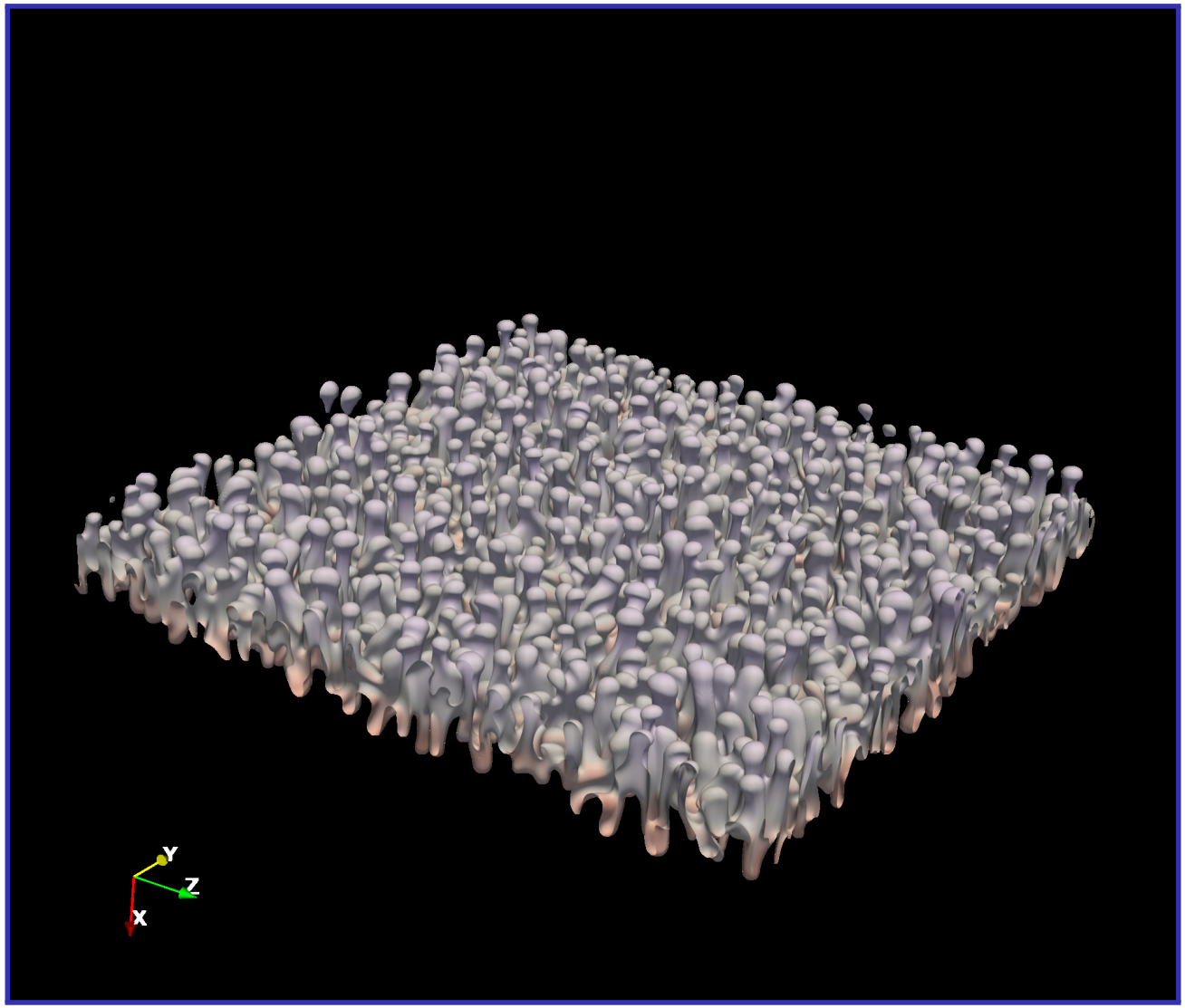
Sediment levelset $C(x, y, z) = 0.5$ at $t = 25$

3D Results - levelset at C=0.5 (lower view, negative z)



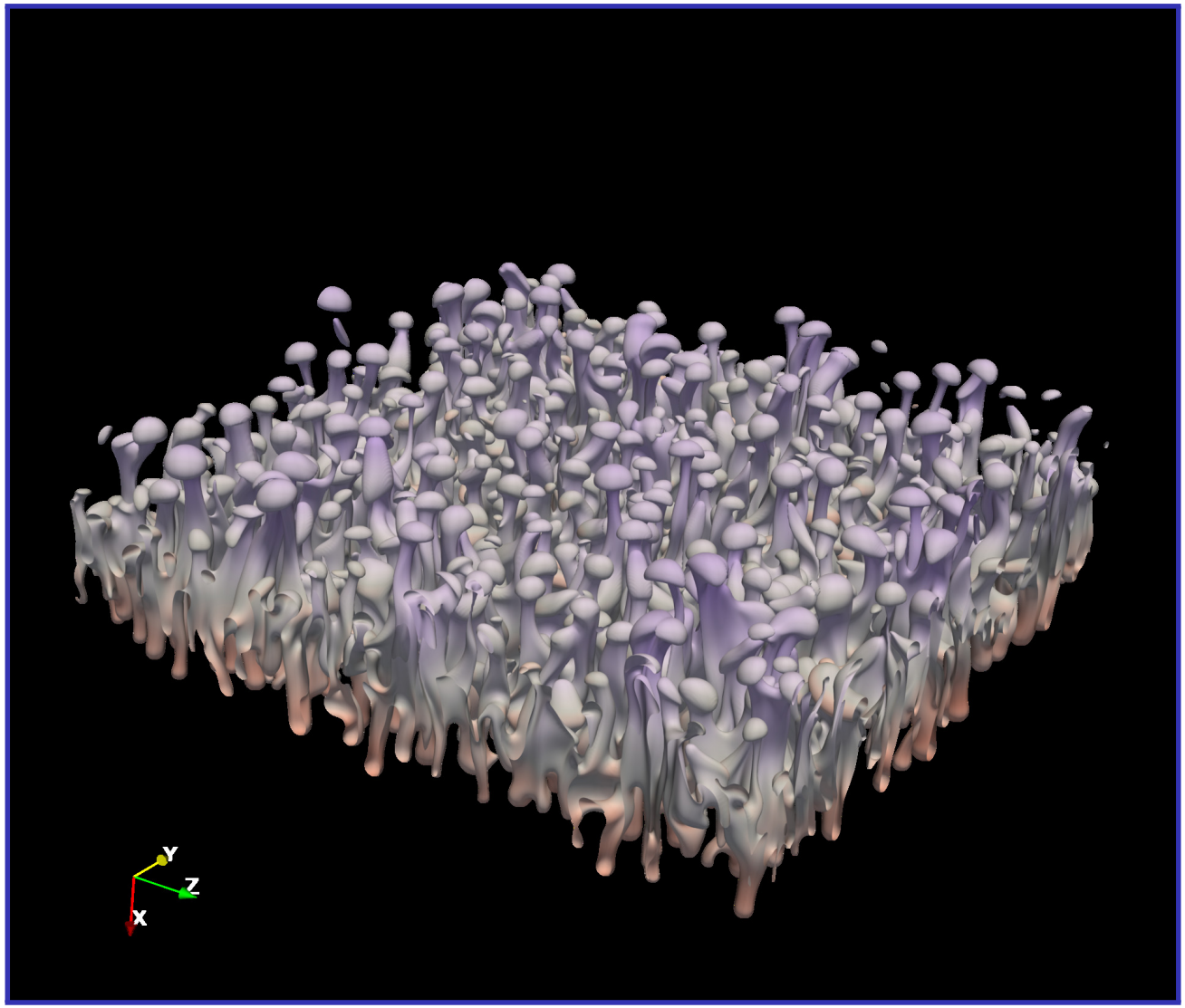
Sediment levelset $C(x, y, z) = 0.5$ at $t = 50$

3D Results - levelset at C=0.5 (lower view, negative z)



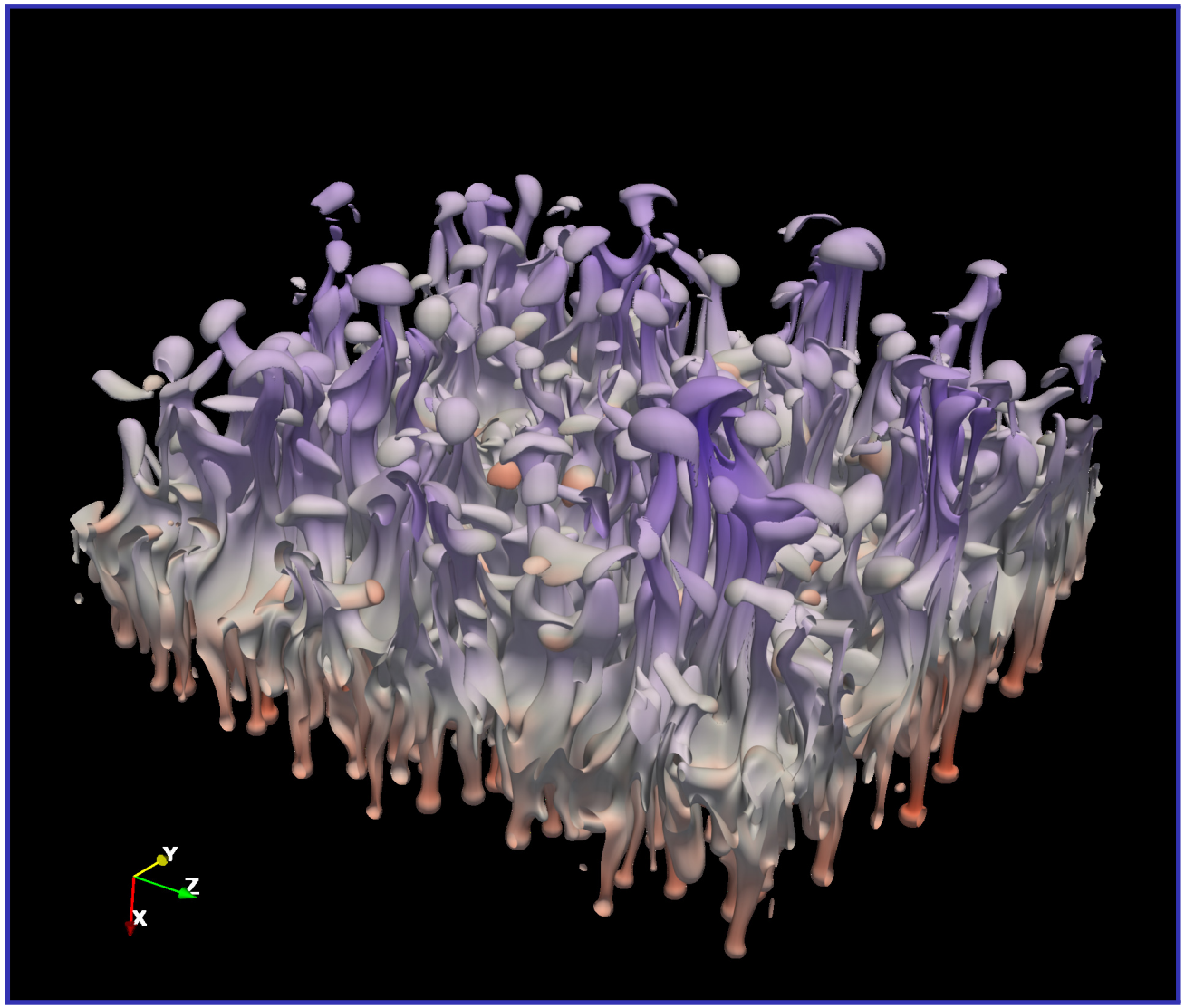
Sediment levelset $C(x, y, z) = 0.5$ at $t = 75$

3D Results - levelset at C=0.5 (lower view, negative z)



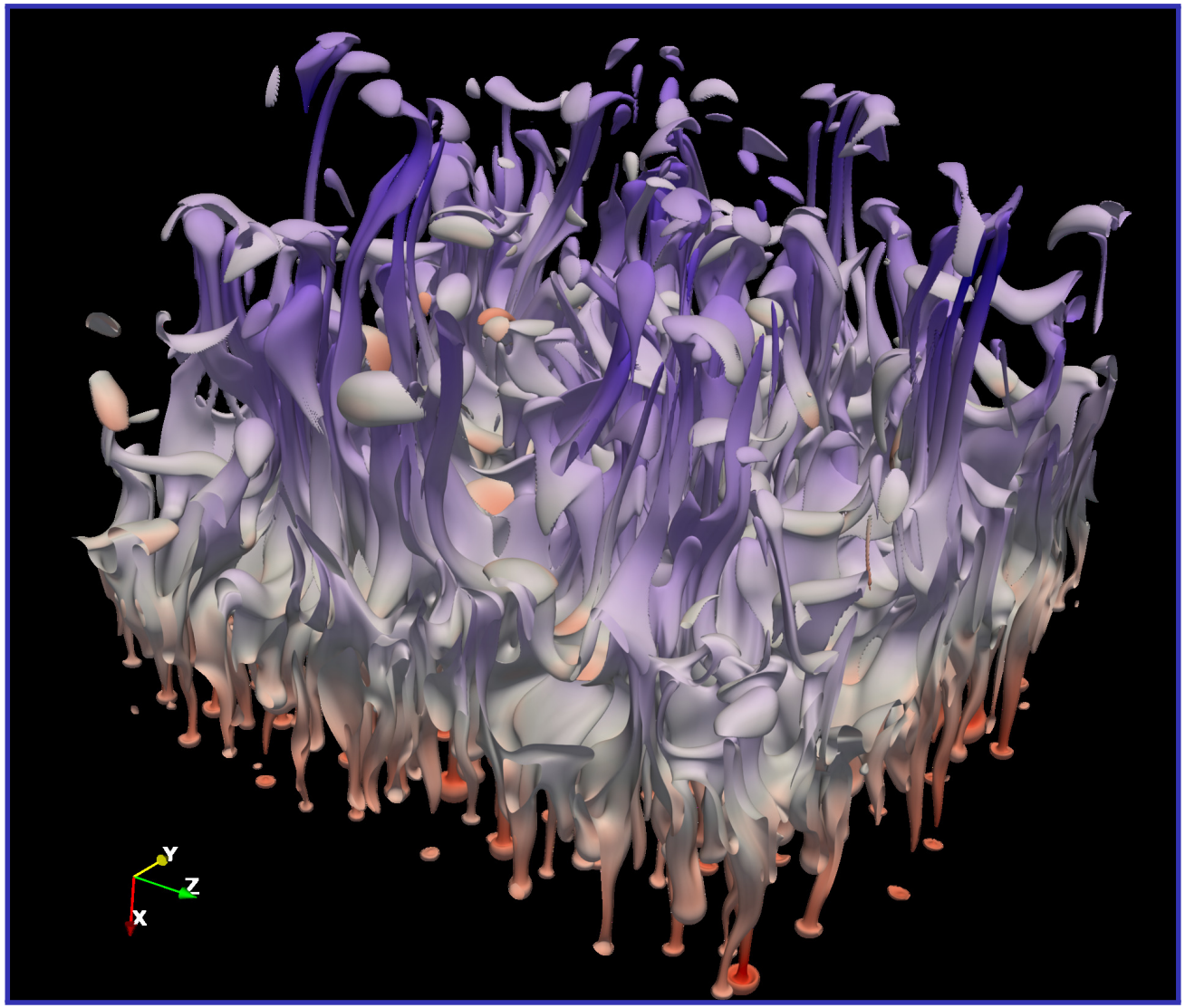
Sediment levelset $C(x, y, z) = 0.5$ at $t = 100$

3D Results - levelset at C=0.5 (lower view, negative z)



Sediment levelset $C(x, y, z) = 0.5$ at $t = 125$

3D Results - levelset at C=0.5 (lower view, negative z)



Sediment levelset $C(x, y, z) = 0.5$ at $t = 150$

3D CPU vs GPU performances (without I/O)

Numerical setup:

- $\Omega = [-110, +65] \times [0, 100]^2, t \in [0, 200]$
- $S_c = 7, \tau = 25, V_p = 0.04, R_s = 2$

Software:

- We use the HySoP library (FDC2, RK4, L4_2, single precision).
- Same OpenCL backend for both CPU and GPU.

Hardware:

- **CPU:** 2 x Intel Xeon E5-2695V4 CPU (18 cores each)
- **GPU:** Nvidia GeForce GTX 980 Ti

Simulation Time		iteration (s)		total (s)	
Resolution	Iterations	CPU	GPU	CPU	GPU
256x64x64	235		0.248		58.47
512x128x128	202		0.701		141.7
1024x256x256	229		2.630		602.4

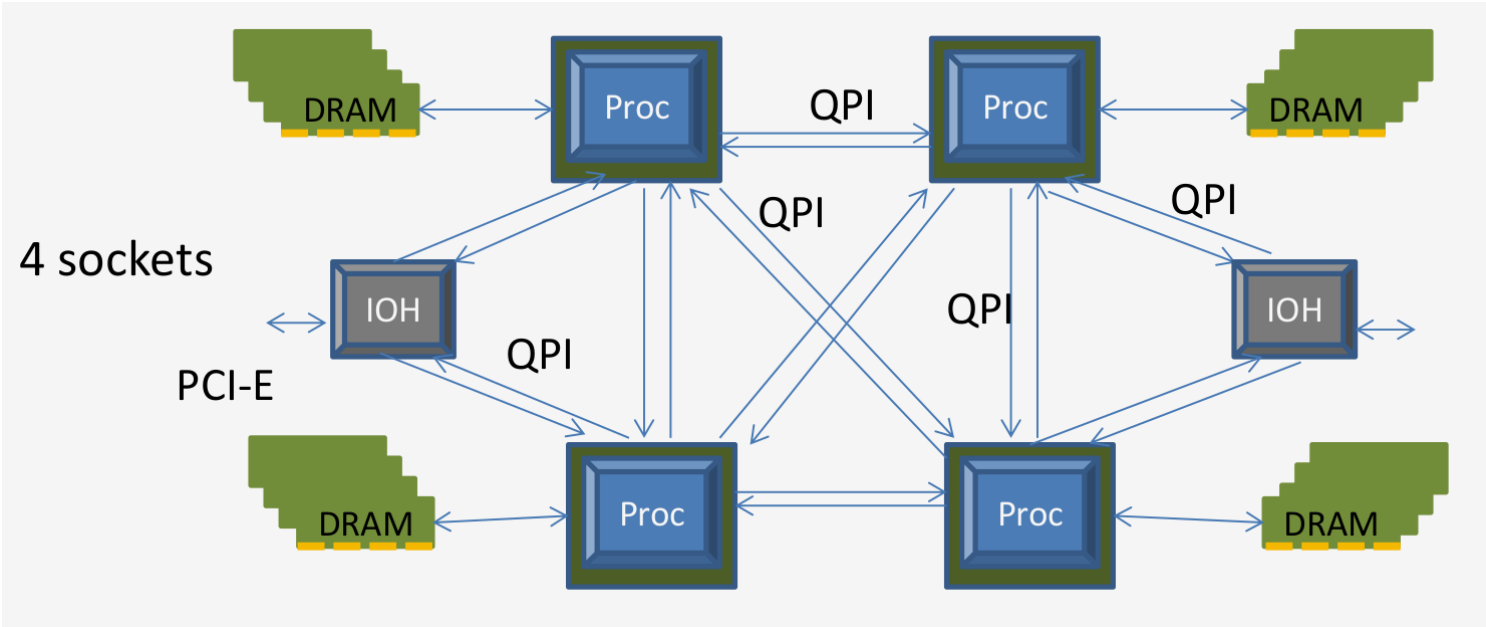
Ongoing 3D simulation

Numerical setup:

- $\Omega = [-110, +65] \times [0, 100]^2, t \in [0, 200]$
- $S_c = 28, \tau = 25, V_p = 0.04, R_s = 2$
- $N = N_x \times N_y \times N_z = 3072 \times 1024 \times 1024$
- 12GB per scalar field

Hardware:

- The simulation requires around 180GB RAM so we run only on CPU...
- Quad socket compute node with 4x18 cores and 512GB RAM.



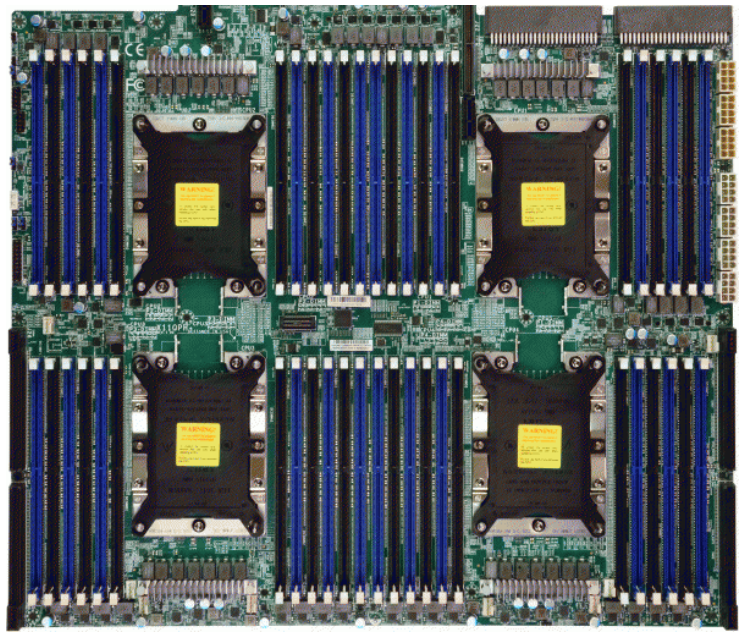
Ongoing 3D simulation

Numerical setup:

- $\Omega = [-110, +65] \times [0, 100]^2, t \in [0, 200]$
- $S_c = 28, \tau = 25, V_p = 0.04, R_s = 2$
- $N = N_x \times N_y \times N_z = 3072 \times 1024 \times 1024$
- 12GB per scalar field

Hardware:

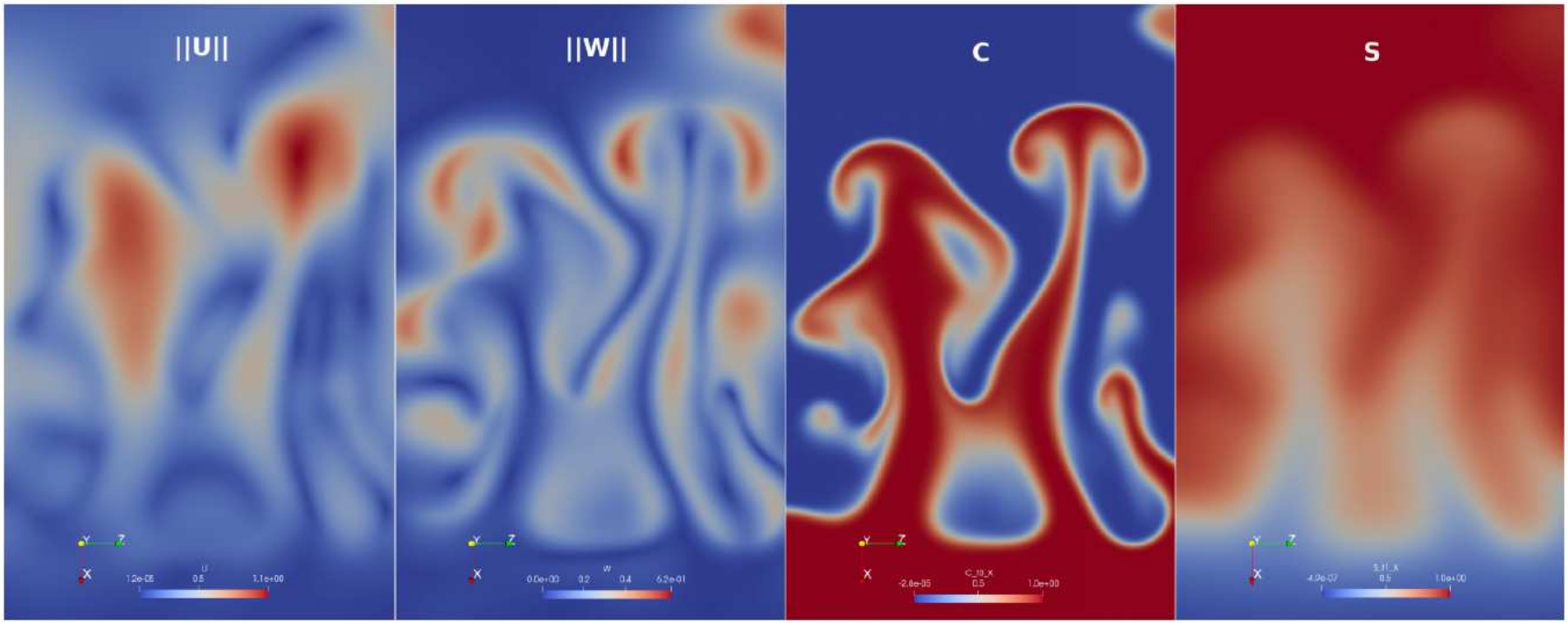
- The simulation requires around 180GB RAM so we run only on CPU...
- Quad socket compute node with 4x18 cores and 512GB RAM.



Multiscale approach

Can we really use a coarse and a fine grid ?

- 3D simulation at reasonable Schmidt number $S_C = 7$.
- With $\tau = 25$ this gives $S_s = \tau S_c = 175$.
- This configuration already exhibits different physical scales:



Conclusion and perspectives

Conclusion

- You can already compute nice simulations with a single compute node (even without `GPU`).
- Our code is in agreement with the results of Meiburg et al. [2]
- We still need to go higher in Schmidt number.

Future developments

- Implement the multi-scale approach and `MPI` FFT-based solvers (global transposition of memory accross all processes).
- Full in-core simulation on multiple GPUs should allow high Schmidt simulations in reasonable time.
- This year we will release `HySoP v2.0` to the public.

References

[1] G. Balarac et al. "Multi-scale Problems, High Performance Computing and Hybrid Numerical Methods". In: *Mathematics for Industry* (2014), pp. 245–255.

[2] P. Burns and E. Meiburg. "Sediment-laden fresh water above salt water: nonlinear simulations". In: *Journal of Fluid Mechanics* 762 (Nov. 2014), pp. 156–195.

[3] Georges-Henri Cottet et al. "High order Semi-Lagrangian particle methods for transport equations: numerical analysis and implementation issues". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 48.4 (July 2014), pp. 1029–1060.

Thanks for your attention !
Any questions ?